An adaptive method for computing invariant manifolds in non-autonomous, three-dimensional dynamical systems

Manuscript submitted to Physica D

An adaptive method for computing invariant manifolds in non-autonomous, three-dimensional dynamical systems

Michał Branicki* and Stephen Wiggins

School of Mathematics, University of Bristol,
Bristol BS8 1TW, United Kingdom

April 23, 2009

Abstract

We present a computational method for determining the geometry of a class of three-dimensional invariant manifolds in non-autonomous (aperiodically time-dependent) dynamical systems. The presented approach can be also applied to analyse the geometry of 3D invariant manifolds in three-dimensional, time-dependent fluid flows. The invariance property of such manifolds requires that, at any fixed time, they are given by surfaces in $\mathbb{R}^3$. We focus on a class of manifolds whose instantaneous geometry is given by orientable surfaces *embedded* in $\mathbb{R}^3$. The presented technique can be employed, in particular, to compute codimension one (invariant) stable and unstable manifolds of hyperbolic trajectories in 3D non-autonomous dynamical systems which are crucial in the Lagrangian transport analysis. The same approach can also be used to determine evolution of a an orientable 'material surface' in a fluid flow. These developments represent the first step towards a non-trivial 3D extension of the so-called *lobe dynamics* – a geometric, invariant-manifold-based framework which has been very successful in the analysis of Lagrangian transport in unsteady, two-dimensional fluid flows. In the developed algorithm, the instantaneous geometry of an invariant manifold is represented by an adaptively evolving triangular mesh with piecewise C^2 interpolating functions. The method employs an automatic mesh refinement which is coupled with adaptive vertex redistribution. A variant of the advancing front technique is used for remeshing, whenever necessary. Such an approach allows for computationally efficient determination of highly convoluted, evolving geometry of codimension one invariant manifolds in unsteady three-dimensional flows. We show that the developed method is capable of providing detailed information on the evolving Lagrangian flow structure in three dimensions over long periods of time, which is crucial for a meaningful 3D transport analysis.

1 Introduction

The geometrical framework of dynamical systems theory has proven to be very useful in the analysis of chaotic transport in unsteady, advection-dominated fluid flows, and resulted in a fundamentally new approach to the subject over the past two decades (e.g. [57, 62, 71, 73] to mention a few). The success of this approach stems largely from the fact that the global space-time topology of trajectories traced by 'fluid particles' in such flows lends itself to geometric analysis in a phase space of an underlying dynamical system which does not depend on the specific form of the system considered. Rather, such a framework allows for identification of certain localised geometrical objects in the fluid flow (considered in the 'space-time' sense) which play a crucial and universal role in organising its global dynamics. When considering redistribution (or transport) of a passive scalar or 'fluid particles' from the Lagrangian perspective, such structures are generally represented by *invariant manifolds* in the dynamical systems terminology. Using the analogy of an inviscid unsteady fluid flow defined over some

*Corresponding author: m.branicki@bristol.ac.uk

time interval $I = [t_i,\ t_f] \subset \mathbb{R}$, invariant manifolds can be thought of as *material* structures composed entirely of trajectories traced by 'fluid particles' evolving with the flow from some (possibly compact) set of 'initial locations' at t_i. One can easily think in this context of a whole menagerie of manifolds satisfying the invariance condition. However, not every flow-invariant manifold provides equally important insight into the global flow structure. In fact, in the case of an arbitrary unsteady fluid flow or, more generally, a flow associated with an arbitrary non-autonomous dynamical system, it is not entirely clear what makes an invariant manifold relevant in transport considerations. One necessary characteristic is that the manifold is *codimension one* which simply means that its dimension is one less than that of the *extended phase space* (i.e. spatial dimensions 'plus' the time dimension) associated with the underlying non-autonomous[1] dynamical system. This is because codimension one manifolds can separate distinct regions in the extended phase space, giving rise to a network of time-dependent *transport barriers* when observed at an ordered sequence of times. It is also known, at least in the case of time-periodic and quasi-periodic time-dependence (e.g. [5, 71]) and for a general 2D non-autonomous case (e.g. [62, 73]), that codimension one stable and unstable manifolds of some 'special' hyperbolic trajectories (discussed later) form a geometrical template, determining the manner in which fluid or tracer particles are dispersed in the flow. Thus, such structures serve as a 'Lagrangian skeleton' for the study of stirring in the inviscid case (often referred to as Lagrangian transport) or mixing in advection-dominated flows (e.g. [5, 57]).

Despite the influence that the dynamical systems approach has had on the analysis of transport processes in many engineering and geophysical flows, the implementations of the invariant-manifold techniques to study Lagrangian transport remain largely restricted, at least in the time-aperiodic case, to the two-dimensional flow configurations. Numerical procedures for determining the geometry of codimension one invariant manifolds in two-dimensional non-autonomous dynamical systems were developed by [51, 49] in the context of Lagrangian transport analysis in two-dimensional unsteady fluid flows. In such a setting, the problem is reduced to determination of two-dimensional manifolds which, at any fixed time, are given by (one-dimensional) curves. When applied to the stable and unstable manifolds of the so-called Distinguished Hyperbolic Trajectories (cf. [32]), the analysis of evolving tangles formed by intersections of these manifolds enabled quantification of the Lagrangian transport in many realistic geophysical flows by means of *lobe dynamics* (e.g. the cross-jet transport in the wind-driven double gyre system [13]; transport across a Balearic front in the Western Mediterranean [48]; or even more complicated mechanisms acting in a oceanic system containing a front which interacts with a set of oceanic eddies [7]).

The computation of codimension one invariant manifolds in three-dimensional aperiodically time dependent case, however much needed, proves to be a considerably tougher problem to tackle than its two-dimensional counterpart and suitable techniques which would be capable of resolving their highly convoluted geometry in a general case do not exist. We are particularly interested in this class of invariant manifolds because it contains the relevant (i.e. codimension one) stable and unstable invariant manifolds of hyperbolic trajectories whose identification is necessary in subsequent Lagrangian transport analysis of unsteady 3D flows. As will be argued later, the flow-invariance requires that such manifolds are represented by a surface[2] for any fixed time. The instantaneous geometry of these surfaces (and of their boundary) changes when followed in time, which requires development of numerical methods that are capable of adaptively tracking these changes throughout the manifold at each examined time instant. This is in stark contrast to the existing methods for computing two-dimensional stable and unstable manifolds of hyperbolic fixed points and hyperbolic orbits in steady 3D flows which, due to the fact that such manifolds are time-independent, can be systematically 'grown outwards' from already computed regions (see the review [39] and references therein). Unfortunately, the drawback of these often sophisticated methods is that they cannot be extended to the time-dependent case (§2.2 for more details). It is also important to note that, although the stable and unstable manifolds of hyperbolic fixed points and orbits in steady 3D flows are

[1] The terms 'non-autonomous', 'time-dependent' or 'unsteady' are essentially synonymous although they are used in different contexts. The term 'non-autonomous' is generally used in connection to a dynamical system which depends on time (in addition to the independent variables which define it), whereas the term 'unsteady' is often used to describe fluid flows which change in time.

[2] This fact might be deceptively obvious to someone with background in fluid dynamics in the context of a 'material surface'. We point out, however, that this is certainly not the only way of embedding a 3-manifold in a $4D$ phase space.

time-independent, this is generally not the case when considering the evolution of an arbitrary material surface even in a steady 3D flow. Finally, we remark here that methods for detection of the so-called *Lagrangian Coherent Structures* in 3D unsteady flows (cf. [44]). These are generally based on identifying surfaces which maximise some measure of hyperbolicity over a finite time interval, usually estimated from finite-time Lyapunov exponent fields (FTLE), which does not apply to a general invariant manifold. Moreover, the relationship (or lack of it) between the LCSs and invariant manifolds is still unclear and deserves a thorough examination which will be addressed in a forthcoming publication.

In this work we describe a numerical technique for approximating a class of three-dimensional invariant manifolds in non-autonomous, three-dimensional dynamical systems. We note that, although our motivation lies in the ability to detect codimension one stable and unstable manifolds of hyperbolic trajectories in such a setting, the developed methods are more general and can be also used to analyse the evolving geometry of orientable C^2 material surfaces (i.e. evolving with the flow). The discussed method, representing an invariant manifold as an ordered sequence of its two-dimensional time 'snapshots', is adaptive in space and allows for detailed, computationally efficient determination of highly convoluted spatio-temporal manifold geometry. It is worth noting here that there are two particular cases where certain important simplifications in the phase space structure can be exploited, leading to more efficient computational approaches. Firstly, when an invariant manifold is time-independent (e.g. the stable and unstable manifolds in autonomous 3D dynamical systems; cf. §2.2) the algorithms developed specifically in this context will offer a better computational efficiency (e.g. [19, 16]). Secondly, if a dynamical system depends periodically on time, the invariant manifold computations (and the associated transport considerations) can be reduced to the study of invariant sets of appropriately constructed Poincaré maps (e.g. [8]). However, the problem of computing invariant manifolds in 3D dynamical systems with aperiodic time dependence cannot be reduced to any of the above cases and it requires a different approach, which we focus on here.

The structure of the paper is as follows. In §2 we develop some analytical notions which are necessary to identify the essential properties of invariant manifolds in non-autonomous dynamical systems. We then show how the invariance properties can be used to represent such manifolds as ordered sequences of their 'fixed-time' snapshots, and we explain how these snapshots are discretised when the underlying invariant manifold is three-dimensional. The remaining discussion is structured around the main building blocks of the algorithm which are sketched in figure 2. In §3 we describe a method used for obtaining the discrete approximations of subsequent manifold snapshots from an initial 'seed' (step 2 in figure 2). (The initial seed can be given, for example, by a material surface at the initial time or by a small patch representing the linear approximation of a stable or unstable manifold of a hyperbolic trajectory at the initial time (cf. Appendix A).) In §4 we show how to determine various geometrical properties of the invariant manifold at any fixed time, which are needed for monitoring the fidelity of its discrete approximation (used in steps 3, 6, 7, 8 of figure 2) represented by a triangular mesh. Techniques used for interpolating this geometrical information between the existing points of the mesh are also discussed there. In §5 we outline methods employed for adapting the triangular mesh to the evolving geometry of the invariant manifold snapshots (see step 8 in figure 2). In §5.3 we describe a procedure for generating the triangular mesh using a variant of the advancing front method, which is performed directly in $\mathbb{R}^3$. This algorithm is used for triangulating the initial manifold snapshot and for subsequent remeshing of the evolving mesh (steps 1 and 7 in figure 2). Finally, in §6 we validate the method against some two-dimensional results and illustrate the performance of the developed algorithm using a few examples of unsteady three-dimensional fluid flows. We finish by summarising the results and discussing a number of outstanding issues for future work in §7. The main challenge in the future work lies in generalisation of the tools developed in the context of *lobe dynamics* in two spatial dimensions to the higher-dimensional case.

2 Theoretical outline of the method

In order to develop a technique capable of numerically approximating the geometry of an *invariant manifold* associated with a system non-autonomous ordinary differential equations, we first need to define what is meant by an *invariant manifold* in such a setting. We develop

the necessary notions in §2.1 and compare the general properties of 'time-dependent' and 'time-independent' invariant manifolds in §2.2. We subsequently show, in §2.3, how the definition of an invariant manifold in a general non-autonomous dynamical system lends itself to an algorithm for computing an approximation of an invariant manifold based on an sequence of its time-sections, which we will refer to hereafter as *manifold snapshots*.

2.1 Invariant manifolds in systems of non-autonomous ODE's

In order to focus the reader's attention, we first summarise the main points discussed in this section. We will eventually define here an invariant manifold (cf. Definition 2.1) of a non-autonomous three-dimensional system of ordinary differential equations defined over a finite time interval as a manifold composed entirely of the system solutions *embedded* [3] in the so-called *extended phase manifold*. The extended phase manifold is in this case four-dimensional and the embeddings of the system solutions represent the system trajectories in this manifold. The manifold invariance is related to its properties under the flow induced by the dynamical system on the extended phase manifold. While such a definition is the most intuitive and is a natural extension of a similar notion from the autonomous setting, it is not the most suitable one for our purposes. We subsequently recast it into a form which is more convenient for our applications (cf. Definition 2.2).

Consider a time-dependent, differentiable (i.e. $C^r,\ r \geqslant 1$) function defined in a connected subset $\Omega \subset I\!R^n$ over a finite time interval I as

$$\boldsymbol{v}(\boldsymbol{x}, t) : \Omega \times I \to I\!R^n, \quad I = [t_i,\ t_f] \subset I\!R, \tag{1}$$

and a system of ordinary differential equations associated with (1) in the form

$$\dot{\boldsymbol{x}} = \boldsymbol{v}(\boldsymbol{x}, t), \qquad \boldsymbol{x} \in \Omega,\ t \in I. \tag{2}$$

Hereafter, we will refer to $\boldsymbol{v}$ as the velocity field. Moreover, if Ω is bounded and $\boldsymbol{p} \in \partial\Omega$ is a point on the smooth boundary $\partial\Omega$, we assume that

$$\boldsymbol{v}(p, t) \cdot \mathbf{n}_{\mathbf{p}} = 0, \quad \forall\ p \in \partial\Omega,\ t \in I, \tag{3}$$

where $\boldsymbol{n}_p$ is a vector normal to $\partial\Omega$ at $\boldsymbol{p}$.

Let ζ be a C^{r+1} injective map

$$\zeta : \quad I \times \Omega \times I \to \Omega, \tag{4}$$

$$\zeta(t, \boldsymbol{x}_0, t_0) = \boldsymbol{x}_0 + \int_{t_0}^{t} \boldsymbol{v}(\zeta(s, \boldsymbol{x}_0, t_0), s)\, \mathrm{d}s, \quad t, t_0 \in I,\ \ \boldsymbol{x}_0 \in \Omega, \tag{5}$$

representing (implicitly) solutions of (2) which we assume here to exist within the whole time interval I. It is then clear from (5) that

$$\zeta(t_0, \boldsymbol{x}_0, t_0) = \boldsymbol{x}_0, \quad \forall\ t_0 \in I. \tag{6}$$

For fixed $\boldsymbol{x}_0$ and t_0, the image of $\zeta(\cdot\,; \boldsymbol{x}_0, t_0)$ is a one dimensional curve embedded in Ω, which is parameterised by t and passes through $\boldsymbol{x}_0$ at time $t = t_0$. Hereafter, we will refer to $\zeta(\cdot\,; \boldsymbol{x}_0, t_0)$ as the Ω-embedded *trajectory* of the system (2). Due to the fact that the system (2) is non-autonomous (i.e. $\partial\boldsymbol{v}/\partial t \neq 0$), the map (4) does not preserve the uniqueness of solutions of (2), since there are infinitely many trajectories in the representation (5) passing through a point $\boldsymbol{x}_0 \in \Omega$ (cf. (6)). For our purposes, it is more convenient to represent the solutions of (2) by embedding them in a higher-dimensional manifold[4], namely $\mathcal{P} = \Omega \times I$, using a map

$$\tilde{\zeta} : \quad I \times \mathcal{P} \to \mathcal{P}, \tag{7}$$

$$\tilde{\zeta}(t; \boldsymbol{x}_0, t_0) = \big(\zeta(t; \boldsymbol{x}_0, t_0), t\big), \quad t, t_0 \in I,\ \ \boldsymbol{x}_0 \in \Omega. \tag{8}$$

[3] If M and N are smooth manifolds, the image of the map $\zeta : M \to N$ is an embedding if ζ is a homeomorphism and the derivative of ζ is everywhere injective.

[4] Formally, $\mathcal{P}$ is not a vector space when $I \subset I\!R$ is finite.

It can be easily verified that the trajectories of (2) embedded in $\mathcal{P}$ do not intersect since $\tilde{\zeta}(t_0;\boldsymbol{x}_0,t_0)=(\boldsymbol{x}_0,t_0)$ and that only one $\mathcal{P}$-embedded trajectory $\tilde{\zeta}(\cdot\,;\boldsymbol{x}_0,t_0)$ passes through the point $(\boldsymbol{x}_0,t_0)\in\mathcal{P}$. This fact, in turn, can be used to construct a one-parameter family of diffeomorphisms, $\{\phi_\tau\}_{\tau\in I}$, given by

$$\phi_\tau:\mathcal{P}\to\mathcal{P},$$

$$\phi_\tau(\boldsymbol{x},t)=(\zeta(\tau,\boldsymbol{x},t),\tau), \tag{9}$$

where $\zeta(\cdot\,,\boldsymbol{x},t)$ is the unique solution of (2) passing through the point $(\boldsymbol{x},t)\in\mathcal{P}$. One may easily check, using the definitions (8) and (9), that every map in the family $\{\phi_\tau\}_{\tau\in I}$ satisfies

$$\phi_{\,t}(\boldsymbol{x},t)=(\boldsymbol{x},t),\quad\forall\,(\boldsymbol{x},t)\in\mathcal{P}. \tag{10}$$

Remark 1. It is important here to bear in mind a rather trivial consequence of the definition (8) that the image of the map $t\to\phi_t(\boldsymbol{x},t)$ represents the entire $\mathcal{P}$-embedded trajectory of (2) passing through $(\boldsymbol{x},t)\in\mathcal{P}$.

In what follows, we will refer to the pair $(\mathcal{P},\{\phi_\tau\}_{\tau\in I})$ as a dynamical system associated with the system of non-autonomous ODE's (2). The manifold $\mathcal{P}$ will be referred to hereafter as a *extended phase manifold* of the dynamical system. The members of the family $\{\phi_\tau\}_{\tau\in I}$, generated by the vector field $\boldsymbol{v}$ (cf. (1)), will be referred to as *evolution maps*.

With the above definitions at hand, we can now define what is meant by an invariant manifold of the system (2).

Definition 2.1 (*Invariant manifold)*. A k-dimensional ($k\leqslant n$) manifold $\mathcal{M}$ embedded in the n-dimensional phase manifold $\mathcal{P}$, i.e. $\mathcal{M}\hookrightarrow\mathcal{P}$, is invariant under the evolution of the dynamical system $\big(\mathcal{P},\{\phi_\tau\}_{\tau\in I}\big)$ associated with the system (2) of ODE's if for every point $(\boldsymbol{x}^{\mathfrak{m}},t^{\mathfrak{m}})=\mathfrak{m}\in\mathcal{M}$ and for all $\tau\in I$, we necessarily have $\phi_\tau(\boldsymbol{x}^{\mathfrak{m}},t^{\mathfrak{m}})\in\mathcal{M}$.

Remarks 2.1

- Based on Remark 1, a manifold $\mathcal{M}\hookrightarrow\mathcal{P}$ is invariant with respect to the dynamics induced by the system (2) if it is 'made up' of entire trajectories of the system (2) embedded in the extended phase manifold $\mathcal{P}$.
- $\mathcal{M}=\mathcal{P}$ is an invariant manifold of the dynamical system $\big(\mathcal{P},\{\phi_\tau\}_{\tau\in I}\big)$.
- Every trajectory $\tilde{\zeta}(\cdot;\boldsymbol{x}_0,t_0)$ is a one dimensional invariant manifold of $\big(\mathcal{P},\{\phi_\tau\}_{\tau\in I}\big)$.

The invariance of $\mathcal{M}$ under the evolution of the dynamical system $\big(\mathcal{P},\{\phi_\tau\}_{\tau\in I}\big)$ imposes restrictions on the way it can be embedded in $\mathcal{P}$ and we will exploit this fact below in order to obtain an alternative definition of an invariant manifold which is better suited for our purpose.

Note first a rather trivial but important consequence of the embedding (8) that, if we define a *time slice* of $\mathcal{P}$ at time t^* as

$$\mathcal{S}_{t^*}=\{(\boldsymbol{x},t)\in\mathcal{P}:\quad t=t^*\in I\}, \tag{11}$$

any $\mathcal{P}$-embedded trajectory of (2) crosses any $\mathcal{S}_{t^*}$ exactly once. We also recall that, based on Remark 1, each $\phi_\tau(\boldsymbol{x},t)$ maps the point $(\boldsymbol{x},t)\in\mathcal{S}_t\subset\mathcal{P}$ to a point in the time slice $\mathcal{S}_\tau$ along a $\mathcal{P}$-embedded trajectory of (2) which passes through $(\boldsymbol{x},t)$. Consequently, the cross-section of any k-dimensional invariant manifold $\mathcal{M}\hookrightarrow\mathcal{P}$ of (2) with $\mathcal{S}_{t^*}$ must be nonempty and have a dimension $k-1$. We can represent such a manifold section in Ω as

$$\mathcal{M}_{t^*}=\mathfrak{P}(\mathcal{M}\cap\mathcal{S}_{t^*}), \tag{12}$$

where the injective map $\mathfrak{P}$ is defined as

$$\mathfrak{P}:\mathcal{P}\to\Omega,$$

$$(\boldsymbol{x},t)\to\boldsymbol{x}. \tag{13}$$

Hereafter, we will refer to $\mathcal{M}_{t^*}$ as the *manifold snapshot* at time t^*.

Using (12) and (13), we can now define an invariant manifold of the system of non-autonomous ODE's (2) in the following way:

Definition 2.2 (*Invariant manifold–alternative definition).* We say that a manifold $\mathcal{M} \hookrightarrow \mathcal{P}$ is invariant under the evolution of the dynamical system $\big(\mathcal{P}, \{\phi_\tau\}_{\tau\in I}\big)$ if

$$\mathfrak{P}\big(\phi_\tau(\mathcal{M}_s, s)\big) = \mathcal{M}_\tau, \quad \forall\ s, \tau \in I. \tag{14}$$

A manifold satisfying (14) is also referred to as an invariant manifold of the system (2). If the manifold $\mathcal{M}$, in addition to (14), is such that $\mathcal{M}_s = \mathcal{M}_\tau$, $\forall\ s, \tau \in I$, we will call it a *time-independent invariant manifold.*

Remarks 2.2

- If the condition (14) is satisfied for every $\boldsymbol{x}_s^{\mathrm{m}} \in \mathcal{M}_s$ and for all $s \in I$, then necessarily $\phi_\tau(\boldsymbol{x}_s^{\mathrm{m}}, s) \in \mathcal{M}$, $\forall\ \tau \in I$. In other words, the manifold $\mathcal{M}$ is invariant according to the definition 2.1. The condition (14) is stronger and it takes into account the fact that, due to the particular embedding of $\mathcal{M}$ in $\mathcal{P}$ (cf. (8)), any ϕ_τ maps points contained in one time slice to another time slice.
- When $I = \mathbb{R}$, the two-parameter family of automorphisms of Ω, $\{\Psi_{\tau s}\}_{\tau \geqslant s}$, such that $\Psi_{\tau s} = \mathfrak{P}\big(\phi_\tau(\cdot, s)\big)$, can be termed a *process* following [64]. The notion of a process is a useful tool in the analysis of 'bifurcation' phenomena in systems of non-autonomous ordinary differential equations (e.g. [41]).

We restrict the following considerations to three-dimensional, invariant manifolds embedded in the four-dimensional phase manifold $\mathcal{P}$ of the dynamical system $\big(\mathcal{P}, \{\phi_\tau\}_{\tau\in I}\big)$; i.e. we assume the domain $\Omega \subset \mathbb{R}^3$ in (1). Consequently, every manifold snapshot $\mathcal{M}_t$, $t \in I$ is a two-dimensional manifold (i.e. a surface). Moreover, when approximating stable and unstable manifolds of hyperbolic trajectories embedded in $\mathcal{P}$, which are of the main interest here, the analysis will be restricted to compact subsets of these manifolds given by two-dimensional *manifolds with boundary.*

Definition 2.3 (*2-manifold with boundary).* A set $M \hookrightarrow \mathbb{R}^3$ is called a C^r 2-manifold with boundary embedded in $\mathbb{R}^3$ if:

1) There exists a collection of open sets $V^\alpha \subset \mathbb{R}^3$, where α belongs to a countable set $\mathscr{A}$, such that if $U^\alpha = V^\alpha \cap M$ then $M = \cup_{\alpha\in\mathscr{A}} U^\alpha$.
2) For each U^α there exists a C^r diffeomorphism $u^\alpha : U^\alpha \to H^2$, where the half-space, $H^2 \subset \mathbb{R}^2$, is defined as
$$H^2 = \{(x_1, x_2) \in \mathbb{R}^2 \ :\ x_1 \geqslant 0\} \tag{15}$$
and its boundary, ∂H^2, is given by
$$\partial H^2 = \{(x_1, x_2) \in H^2 \ :\ \ x_1 = 0\}. \tag{16}$$
3) If $U^\alpha \cap U^\beta \neq \emptyset$, then $u^\alpha \circ (u^\beta)^{-1}$ and $u^\beta \circ (u^\alpha)^{-1}$ are C^r diffeomorphisms .

The boundary of M, denoted by ∂M, is a set of points that are inverse images of ∂H^2 under some chart (U^α, u^α). If $M - \partial M$ is a C^r 2-manifold embedded in $\mathbb{R}^3$, then ∂M is a one-dimensional C^r manifold (i.e. a C^r curve) embedded in $\mathbb{R}^3$ (see, for example, [43] for more details).

Remark 2.3 A term *material surface* is often used in fluid dynamical considerations to refer to a 2-manifold whose evolution is completely determined by the velocity of the fluid. In other words, if M_s is a material surface at some time s, then $\mathfrak{P}\big(\phi_\tau(M_s, s)\big)$ represents the geometry of the same material surface at time τ. Therefore, based on Definition 2.2, M_s can be considered as a snapshot of an invariant 3-manifold associated with the flow. Moreover, if M_s is a 2-manifold with boundary ∂M_s given by some C^r curve, then $\mathfrak{P}\big(\phi_\tau(M_s, s)\big)$ is a 2-manifold with boundary given by $\mathfrak{P}\big(\phi_\tau(\partial M_s, s)\big)$.

2.2 Time-independent invariant manifolds

As stated in Definition 2.1, an invariant manifold $\mathcal{M}$ of the system (2) is time-independent if all of its snapshots (cf. (12)) are identical, i.e.

$$\mathcal{M}_s = \mathcal{M}_t, \ \ \forall\ s, t \in I. \tag{17}$$

Time-independent manifolds are particularly important in systems of autonomous ODE's (i.e. when $\partial \boldsymbol{v}(\boldsymbol{x},t)/\partial t \equiv 0$ in (2)), because the stable and unstable invariant manifolds of hyperbolic fixed points, and of hyperbolic trajectories, are of this type (see, for example, [72]).

Clearly, if $\mathcal{M} \hookrightarrow \mathcal{P}$ is time-independent, the knowledge of the geometry of a single snapshot, say $\mathcal{M}_{t^*}$, is sufficient to determine, if necessary, the geometry of the whole invariant manifold (since in such a case $\mathcal{M} = \mathcal{M}_{t^*} \times I$). However, in applications to autonomous vector fields, it is more convenient and appropriate[5] to consider the dynamics in $\Omega \subset I\!R^3$ rather than $\mathcal{P}\,(= \Omega \times I)$ in which case the invariant manifold can be identified with its snapshot $\mathcal{M}_{t^*}$ in the following way.

Note that if $\mathcal{M}$ is a time-independent invariant manifold of (2) then, based on (14), we have

$$\mathfrak{P}\big(\phi_\tau(\mathcal{M}_{t^*}, s)\big) = \mathcal{M}_{t^*}, \quad \forall\, s, \tau \in I, \tag{18}$$

which implies that the (arbitrary) snapshot $\mathcal{M}_{t^*}$ is invariant with respect to the map

$$\mathfrak{J}_{\tau s} : \Omega \to \Omega, \quad s, \tau \in I, \tag{19}$$

$$\boldsymbol{x} \to \mathfrak{P}\big(\phi_\tau(\boldsymbol{x}\,, s)\big). \tag{20}$$

Furthermore, based on the definitions of $\mathcal{P}$ and ϕ_τ (cf. (13), (9)), one can easily see that

$$\mathfrak{J}_{\tau s}(\boldsymbol{x}) = \zeta(\tau; \boldsymbol{x}, s), \quad \mathfrak{J}_{\tau\tau}(\boldsymbol{x}) = \boldsymbol{x}, \tag{21}$$

where $\zeta(\cdot\,; \mathfrak{m}, s)$ is an Ω-embedded trajectory of (2) passing through $\boldsymbol{x} \in \Omega$ (see (5)). Note here that if we additionally assume $I = I\!R$ and require that $\tau > s$, the family $\{\mathfrak{J}_{\tau s}\}_{\tau \geqslant s}$ becomes a *process* mentioned earlier in Remarks 2.2 . Moreover, if $\partial \boldsymbol{v}(\boldsymbol{x},t)/\partial t \equiv 0$ in (2), it can be seen from (5) that such a process reduces to the 'usual' flow, $\Phi_\tau : \Omega \to \Omega$, $\tau \in I\!R$, of the vector field $\boldsymbol{v}$ on Ω.

Two important conclusions can be drawn immediately from the invariance properties (17) and (18), respectively:

If $\mathcal{M} \hookrightarrow \mathcal{P}$ is a time-independent invariant manifold of (2) and $\mathfrak{m} \in \mathcal{M}_{t^*}$ then

(i) $\mathfrak{m} \in \mathcal{M}_t$ for all $t \in I$.

(ii) any Ω-embedded trajectory (5) of the system (2), passing through $\mathfrak{m}$, is contained entirely in $\mathcal{M}_{t^*}$. In other words, if a point $\mathfrak{m} \in \Omega$ is such that

$$\mathfrak{m} \in \mathcal{M}_{t^*}, \text{ then also } \quad \mathfrak{J}_{\tau s}(\mathfrak{m}) \in \mathcal{M}_{t^*}, \ \forall\, s, \tau \in I.$$

Therefore, $\mathcal{M} = \mathcal{M}_{t^*} \times I \hookrightarrow \mathcal{P}$ is a time-independent invariant manifold of the system (2) if $\mathcal{M}_{t^*}$ is an invariant set of the family $\{\mathfrak{J}_{\tau s}\}_{\tau, s \in I}$. Consequently, any subset $\sigma \subset \mathcal{M}_{t^*}$ has to be determined only once. Furthermore, if the subset σ is such that no trajectory ζ of (2) is tangent to the boundary, $\partial\sigma$, the geometry of the snapshot $\mathcal{M}_{t^*}$ exterior to σ can be determined, in principle, by identifying trajectories of (2) which pass through points $\mathfrak{m} \in \sigma$.

There exists a number of sophisticated numerical methods computing stable and unstable manifolds of autonomous vector fields which exploit the above properties of time-independent invariant manifolds (see [39, 19] and references therein). However, due to the fact that these algorithms essentially rely on the invariance of the manifold snapshots (via the relations (17) and (18)), they cannot be extended to the general non-autonomous setting when $\mathcal{M}_t \neq \mathcal{M}_s$ (if $t \neq s$). In such a case, different techniques have to be developed in order to determine the geometry of each manifold snapshot independently.

2.3 Mesh approximation of an invariant manifold

Generally, the geometry of an invariant manifold $\mathcal{M} \hookrightarrow \mathcal{P}$ cannot be determined analytically and, at best, an approximation of the manifold in a suitable numerical representation can be obtained. Consider first a two-dimensional snapshot, $\mathcal{M}_{t_n}$ (cf. (12)), of the invariant manifold at some time instant t_n which is represented numerically by a pair $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$, referred to hereafter as the *mesh* approximating $\mathcal{M}_{t_n}$ (see figure 1a). $\mathfrak{M}_{t_n}$ denotes here a discrete, ordered and finite set of *mesh vertices* which are contained in $\mathcal{M}_{t_n}$, i.e.

$$\mathfrak{M}_{t_n} = \{\mathfrak{m}^i : \mathfrak{m}^i \in \mathcal{M}_{t_n}, \ \ i = 1, \ldots, M\}. \tag{22}$$

[5] See the discussion at the beginning of §2.1

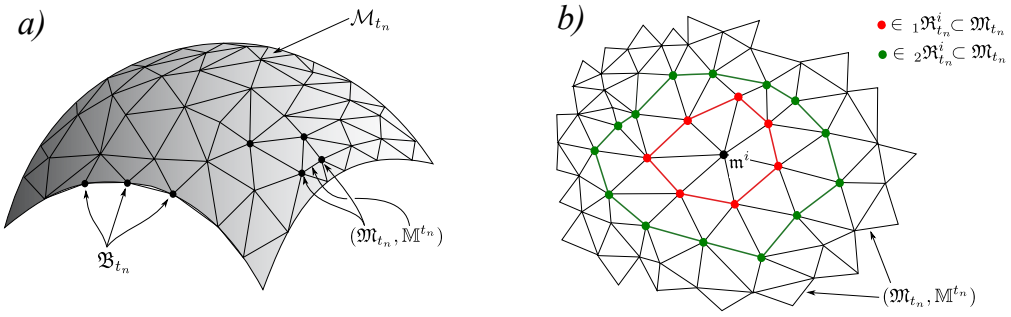

Figure 1: Schematic illustration of the notions introduced in §2.3. a) triangular mesh, denoted by $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$, which is embedded in $\Omega \subset I\!R^3$ and approximates the manifold snapshot $\mathcal{M}_{t_n}$ (cf. 12). The mesh contains information about locations of its vertices, i.e. $\mathfrak{M}_{t_n}$ (22), and connections between them, $\mathbb{M}_{t_n}$ (23). b) definition of the first ring of vertices, ${}_1\mathfrak{R}^i_{t_n}$, (red) and of the second ring, ${}_2\mathfrak{R}^{\mathrm{i}}{}_{t_n}$, (green) of the mesh vertex $\mathfrak{m}^i$.

Information about connectivity between the mesh vertices is contained in $\mathbb{M}^{t_n}$ which is represented in the form of a sparse, symmetric matrix of dimension $M \times M$, such that

$$\mathbb{M}_{jk} = \begin{cases} 1 & \text{if the vertex } \mathfrak{m}^j \text{ is connected to the vertex } \mathfrak{m}^k, \\ 0 & \text{if the vertex } \mathfrak{m}^j \text{ is not connected to the vertex } \mathfrak{m}^k. \end{cases} \tag{23}$$

In the following considerations we will use a *triangular mesh* for which, in addition to (23), if $\mathbb{M}_{ij} = 1$ and $\mathbb{M}_{ik} = 1$, then necessarily $\mathbb{M}_{kj} = 1$.

If the considered manifold has a boundary, i.e. $\partial\mathcal{M}_{t_n} \neq \emptyset$ (see Definition 2.3), we will refer to mesh vertices contained in the set

$$\mathfrak{B}_{t_n} = \{\mathfrak{m} \in \mathfrak{M}_{t_n} : \quad \mathfrak{m} \in \partial\mathcal{M}_{t_n}\}, \tag{24}$$

as the *boundary vertices* of the mesh $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$.

For any mesh vertex $\mathfrak{m}^i \in \mathfrak{M}_{t_n}$, we will refer to all mesh vertices directly connected to $\mathfrak{m}^i$ as its first ring neighbours (see figure 1b), i.e.

$${}_1\mathfrak{R}^i_{t_n} = \{\mathfrak{m}^j \in \mathfrak{M}_{t_n} : \quad \mathbb{M}^{t_n}_{ij} \neq 0\}, \tag{25}$$

and we denote the set of indices of manifold points $\mathfrak{m}^j$ contained in ${}_1\mathfrak{R}^i_{t_n}$ by $\mathrm{Idx}[{}_1\mathfrak{R}^i_{t_n}]$, i.e.

$$\mathrm{Idx}[{}_1\mathfrak{R}^i_{t_n}] = \{j \in \mathbb{Z}^+ : \mathfrak{m}^j \in {}_1\mathfrak{R}^i_{t_n}\}. \tag{26}$$

The second ring of neighbours of $\mathfrak{m}^i \in \mathfrak{M}_{t_n}$, denoted by ${}_2\mathfrak{R}^i_{t_n}$, is defined as

$${}_2\mathfrak{R}^{\mathrm{i}}{}_{t_n} = \{\mathfrak{m}^k \in \mathfrak{M}_{t_n} \backslash ({}_1\mathfrak{R}^i_{t_n} \cup \mathfrak{m}^i) : \quad \mathbb{M}^{t_n}_{jk} \neq 0, \ j \in \mathrm{Idx}[{}_1\mathfrak{R}^i_{t_n}]\}, \tag{27}$$

i.e. ${}_2\mathfrak{R}^i_{t_n}$ is the union of first-ring neighbours of the vertices contained in ${}_1\mathfrak{R}^i_{t_n}$, excluding the vertices already in ${}_1\mathfrak{R}^i_{t_n}$ (cf. figure 1b). The subsequent rings are constructed in an iterative fashion, following the structure of (25) and (27).

3 Flow-induced mesh advection

In applications of invariant manifold methods to Lagrangian transport problems, it is convenient to represent an invariant manifold, $\mathcal{M} \hookrightarrow \mathcal{P}$, associated with (2) as an ordered sequence of manifold snapshots, $\{\mathcal{M}_{t_n}\}_{n\in\mathbb{Z}_N}$, $\mathcal{M}_{t_n} \subset \Omega \subset I\!R^3$ (see (12)), corresponding to a sequence of 'observation' times

$$t_0 < t_1 < t_2 < \cdots < t_n < t_{n+1} < \ldots, \tag{28}$$

such that each $t_n \in I$; we will denote this sequence as $\{t_n\}_{n\in\mathbb{Z}_N}$; $Z_N = 0, \ldots, N$. The subsequent manifold snapshots in the sequence are related to each other via (14). Consequently,

based on Definition‘2.1, a mesh approximating the manifold snapshot $\mathcal{M}_{t_{n+1}}$ can be obtained from the mesh $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$ by ‘advecting’ it to the next observation time along trajectories of the system (2), i.e.

$$\big(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}}\big) = \big(\mathfrak{P}\big(\phi_{t_{n+1}}(\mathfrak{M}_{t_n}, t_n)\big), \mathbb{M}^{t_n}\big). \tag{29}$$

Thus, in principle, one can approximate the sequence of snapshots, $\{\mathcal{M}_{t_n}\}_{n\in\mathbb{Z}_N}$, of the invariant manifold $\mathcal{M}$ by the sequence $\big\{(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})\big\}_{n\in\mathbb{Z}_N}$, where the subsequent meshes are obtained from an initial mesh, $(\mathfrak{M}_{t_0}, \mathbb{M}^{t_0})$, by means of (29). Clearly, such an iterative procedure relies on the knowledge of the initial manifold snapshot, $\mathcal{M}_{t_0}$, and its mesh approximation. As we show in §6, in the case of computing stable and unstable manifolds of hyperbolic trajectories the initial manifold snapshot is given by a small disc contained in either the stable or unstable subspaces of a linearised vector field (1) about the hyperbolic trajectory at t_0. Numerically, the ‘advection step’ can be achieved by integrating the trajectories of the system (2) with the help of one of many available ODE solvers. Here, we use the MATLAB’s variable order, multistep and time-adaptive solver, ‘ode15s’, which uses the so-called Numerical Differentiation Formulas (NDF’s) to perform this task. The method is fully implicit and A-stable (for more information see MATLAB’s online help and `www.mathworks.com/access/helpdesk/help/pdf_doc/otherdocs/ode_suite.pdf`).

The mesh obtained simply by mapping vertices contained in $\mathfrak{M}_{t_n}$ to the slice of the next ‘observation’ time, $\mathcal{S}_{t_{n+1}}$, does not generally represent a good approximation of $\mathcal{M}_{t_{n+1}}$. This is because the map $\phi_{t_{n+1}}$, which advects and deforms the mesh $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$, does not preserve the optimal distribution of mesh vertices in the sense that regions of high concentration of vertices in the advected mesh (inherited from the ‘old’ mesh $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$) may not coincide with regions of high curvature on $\mathcal{M}_{t_{n+1}}$. The algorithm, which we describe in the following sections, combines the advection of the mesh vertices to the next time slice (see stage (2) in figure 2) with appropriate refinement techniques (cf. §5.1, §5.2) which modify, if necessary, the vertex distribution in the subsequent meshes.

3.1 Advection of the boundary

If the snapshots of the invariant manifold have a boundary, the geometry of each subsequent $\partial\mathcal{M}_{t_{n+1}}$ can be estimated from the discrete set of boundary vertices, $\mathfrak{B}_{t_{n+1}}$, of the advected mesh at t_{n+1}, i.e. $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}_{t_{n+1}})$. Such an approach provides, in principle, an approximation of each (closed) boundary curve, which is of the same order as that for the remaining part of the manifold snapshot $\mathcal{M}_{t_{n+1}}$. However, due to the fact that for every $\mathfrak{m}^i \in \mathfrak{B}_{t_{n+1}}$ the distribution of neighbouring vertices in the mesh $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}})$ is highly anisotropic (i.e. there are no vertices on the ‘other’ side of the boundary), the mesh refining procedures (cf. §5.1) are more sensitive to numerical errors (see §4.3) which may accumulate after a sequence of advection steps. These effects become particularly important when initiating the remeshing algorithm (cf. §5.3) which requires accurate estimates of the geodesic curvature (42) of the boundary $\partial\mathcal{M}_{t_{n+1}}$ which is usually subjected to strong deformation. Therefore, apart from advecting the mesh between the sequence of observation times $\{t_n\}_{n\in\mathbb{Z}_N} \subset I$, we also advect an independent copy of the boundary, denoted by $\mathfrak{B}_{t_n}$. Similarly, when the manifold snapshots $\mathcal{M}_{t_{n+1}}$ are given by closed surfaces, a closed curve $\mathfrak{L}_{t_{n+1}}$ contained in $\mathcal{M}_{t_{n+1}}$ is needed for initiating the remeshing procedure (the role of $\mathfrak{L}_{t_{n+1}}$ is analogous to $\mathfrak{B}_{t_{n+1}}$ during the remeshing, except that the mesh is constructed on both ‘sides’ of $\mathfrak{L}_{t_n}$). The sequence of $\{\mathfrak{B}_{t_n}\}_{n\in\mathbb{Z}_N}$ (or $\{\mathfrak{L}_{t_n}\}_{n\in\mathbb{Z}_N}$) is determined using an algorithm which employs similar techniques to those used previously for computing evolution of one-dimensional snapshots of 2D invariant manifolds in unsteady, 2D fluid flows [51]. Details of this algorithm are given in Appendix B. Similarly to $\mathfrak{B}_{t_n}$, each boundary curve $\mathfrak{B}_{t_n}$ is numerically approximated in the algorithm by an ordered set of connected points, but the resolution of $\{\mathfrak{B}_{t_n}\}_{n\in\mathbb{Z}_N}$ is maintained at much higher levels than that of $\{\mathfrak{B}_{t_n}\}_{n\in\mathbb{Z}_N}$.

4 Estimation of local manifold properties from its discrete representation

In order to monitor the fidelity of the sequence of meshes (29) which approximate the sequence of snapshots $\{\mathcal{M}_{t_n}\}_{n\in\mathbb{Z}_N}$ of the invariant manifold $\mathcal{M}$, one needs to be able to estimate certain differential properties of each snapshot $\mathcal{M}_{t_n}$ from its piecewise-linear approximation given

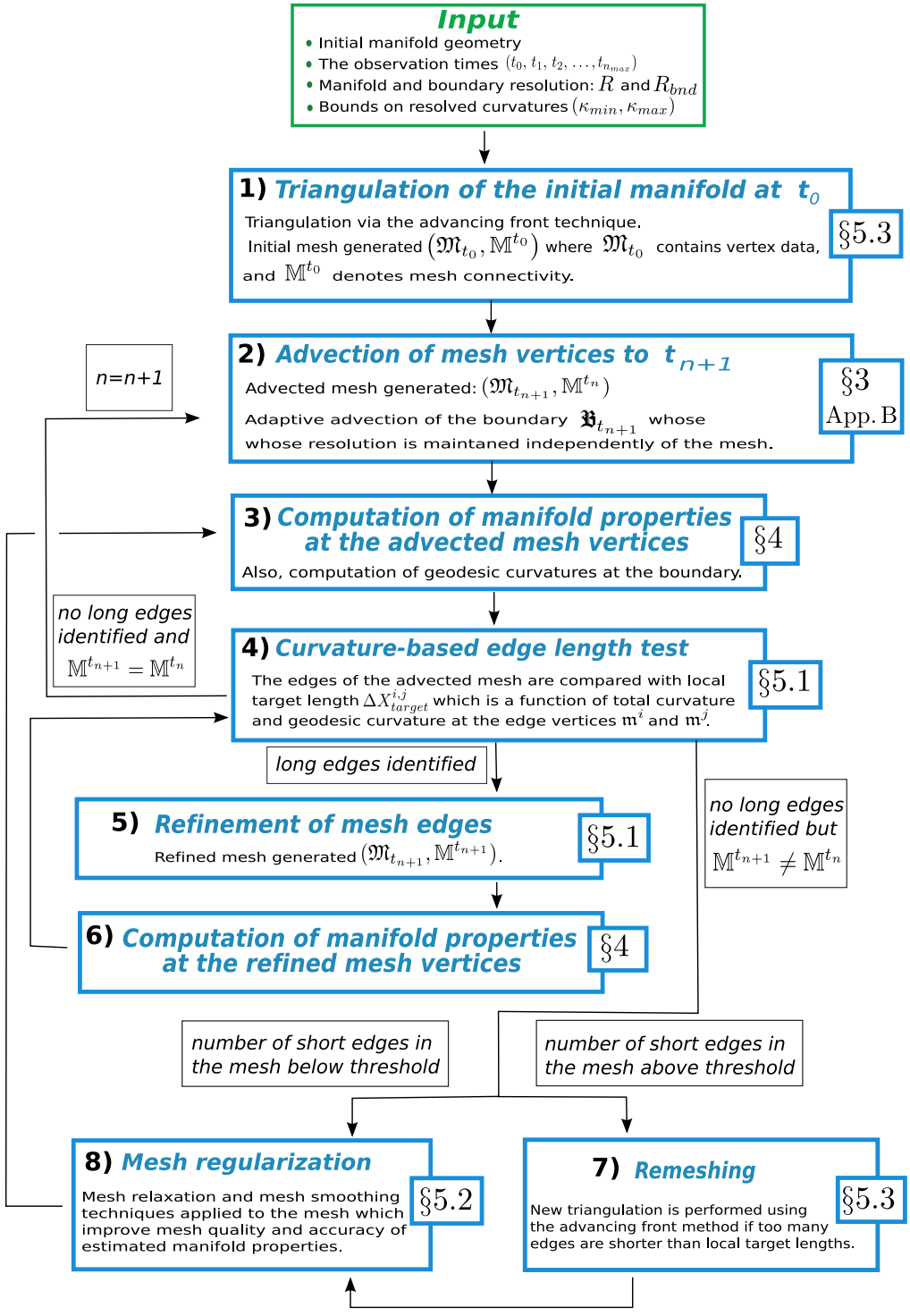

Figure 2: A simplified diagram showing the structure of the algorithm for computing 3D invariant manifolds of non-autonomous dynamical systems. The data flow is marked by the arrows. The sections of the paper in which the respective parts of the algorithm are discussed are indicated on the right edges of the boxes shown in the diagram.

by the triangular mesh $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$. In this section we outline procedures which are used in our algorithm (steps 3, 6 in figure 2) to accomplish these tasks. First, we recapitulate some necessary analytical notions from differential geometry of surfaces (§4.1) which are later used to construct an algorithm for estimating manifold curvatures and normals at the mesh vertices (§4.3). Finally, a method used in the algorithm for interpolation of new manifold points between the mesh vertices is outlined in §4.4 (steps 5, 7, 8 in figure 2).

4.1 Necessary analytical definitions

As noted earlier in §2.1 (see (12)), each snapshot $\mathcal{M}_t$ of the invariant manifold $\mathcal{M}$ is embedded in $\Omega \subset I\!R^3$ (cf. (12)). Therefore, every point $\mathfrak{m} \in \mathcal{M}_t$ can be represented in the Cartesian coordinates defined on $I\!R^3$ which is formally expressed by restricting the canonical Cartesian coordinate chart to $\mathcal{M}_t \cap \Omega$, i.e.

$$\mathcal{E}|_{\mathcal{M}_t \cap I\!R^3} : \mathcal{M}_t \cap \Omega \to I\!R^3$$

$$\mathfrak{m} \to \begin{pmatrix} x^{\mathfrak{m}} \\ y^{\mathfrak{m}} \\ z^{\mathfrak{m}} \end{pmatrix} \equiv \boldsymbol{X}^{\mathfrak{m}}. \tag{30}$$

We will refer to $x^{\mathfrak{m}}$, $y^{\mathfrak{m}}$ and $z^{\mathfrak{m}}$ as the Cartesian coordinates of the manifold point $\mathfrak{m}$.

Since we assumed that each $\mathcal{M}_t$ is a two-dimensional differentiable manifold (§2.1), for every $\mathfrak{m} \in \mathcal{M}_t$ there exists a neighbourhood, $\sigma \subset \mathcal{M}_t$, containing $\mathfrak{m}$ and a local coordinate chart, $(\sigma, \mathcal{L}_\sigma)$, such that the diffeomorphism

$$\mathcal{L}_\sigma : \mathcal{M}_t \supset \sigma \to I\!R^2, \tag{31}$$

transforms points contained in σ into a subset of $I\!R^2$, i.e.

$$\mathcal{L}_\sigma(\mathfrak{m}) = (\xi_\sigma^{\mathfrak{m}}\ ,\ \eta_\sigma^{\mathfrak{m}}) \in I\!R^2, \qquad \forall\ \mathfrak{m} \in \sigma. \tag{32}$$

We will refer to $\mathcal{L}_\sigma$ as the local coordinate system on σ with coordinates $(\xi_\sigma, \eta_\sigma)$. The local coordinate charts are consistent in the sense that the transition maps between overlapping regions of manifold patches are also diffeomorphisms, i.e. if $\delta \cap \sigma \neq \emptyset$, the maps

$$\mathcal{L}_\sigma \circ \mathcal{L}_\delta^{-1}|_{\mathcal{L}_\delta(\delta \cap \sigma)} : \mathcal{L}_\delta(\delta \cap \sigma) \to \mathcal{L}_\sigma(\delta \cap \sigma). \tag{33}$$

Using the local coordinates $(\xi_\sigma, \eta_\sigma)$, each point σ can be represented in $I\!R^3$ by means of the parameterisation

$$\boldsymbol{X}(\xi_\sigma, \eta_\sigma) = \mathcal{E} \circ \mathcal{L}_\sigma^{-1}(\xi_\sigma\ ,\ \eta_\sigma) = \begin{pmatrix} x(\xi_\sigma, \eta_\sigma) \\ y(\xi_\sigma, \eta_\sigma) \\ z(\xi_\sigma, \eta_\sigma) \end{pmatrix}, \qquad (\xi_\sigma, \eta_\sigma) \in \mathcal{L}_\sigma(\sigma), \tag{34}$$

which implies that the local surface characteristics, in particular surface curvature and surface normal, at any point of the manifold can be determined using the coefficients of the first and the second fundamental forms (see, for example, [28, §16.6] or any other book on differential geometry of surfaces).

Assume that a manifold patch $\sigma \subset \mathcal{M}_t$ is parameterised by $(\xi_\sigma, \eta_\sigma)$. Then, the surface normal at any point $\mathfrak{m} \in \sigma$ with coordinates (32) is given by

$$\boldsymbol{n}^{\mathfrak{m}} = \boldsymbol{X}_\xi^{\mathfrak{m}} \wedge \boldsymbol{X}_\eta^{\mathfrak{m}} / J, \tag{35}$$

where $\boldsymbol{X}_\xi^{\mathfrak{m}} = \partial \boldsymbol{X} / \partial \xi_\sigma|_{(\xi_\sigma^{\mathfrak{m}}, \eta_\sigma^{\mathfrak{m}})}$, $\boldsymbol{X}_\eta^{\mathfrak{m}} = \partial \boldsymbol{X} / \partial \eta_\sigma|_{(\xi_\sigma^{\mathfrak{m}}, \eta_\sigma^{\mathfrak{m}})}$ and the Jacobian, J, can be written as

$$J = |\boldsymbol{X}_\xi^{\mathfrak{m}} \wedge \boldsymbol{X}_\eta^{\mathfrak{m}}| = \sqrt{EG - F^2}. \tag{36}$$

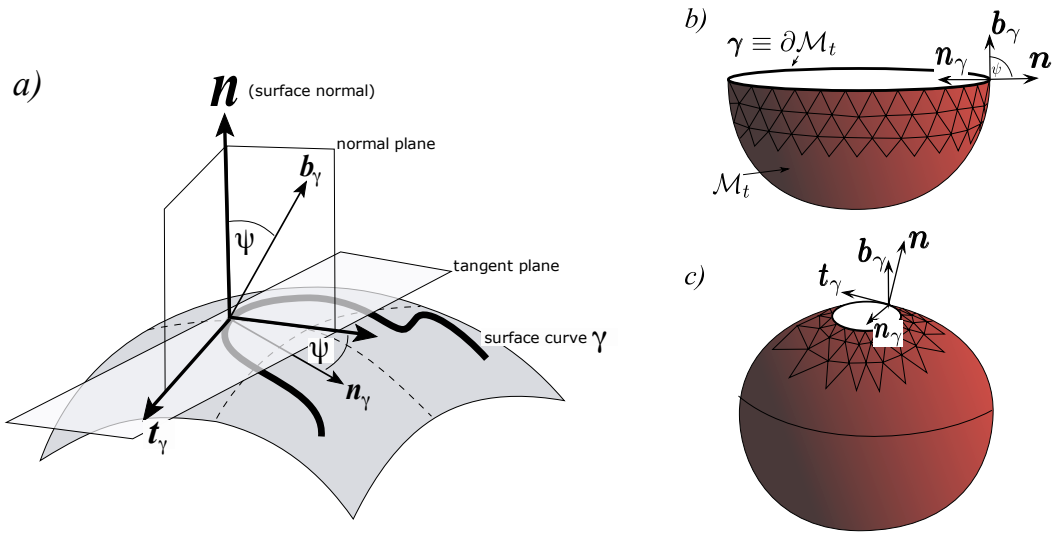

Figure 3: The importance of monitoring the geodesic curvature when the the snapshots $\mathcal{M}_{t_n}$ are given by manifolds with boundary (cf. Definition 2.3). a) a surface curve $\boldsymbol{\gamma}$ and its Frenet frame; the geodesic curvature κ_γ (42) takes into account the angle ψ between the surface normal $\boldsymbol{n}$ and the curve binormal $\boldsymbol{b}_\gamma$. b) sketch of a situation when the geodesic curvature of the boundary is not important ($\kappa_\gamma = 0$) and in c) the geodesic curvature information is important in properly resolving the boundary (i.e. the triangle size of the mesh resolving the interior of the manifold snapshot is too large for approximating its boundary to the same degree).

The coefficients of the first fundamental form, E, F, and G, are given by

$$\begin{aligned} E &= \langle \boldsymbol{X}^{\mathfrak{m}}_{\xi}, \boldsymbol{X}^{\mathfrak{m}}_{\xi} \rangle, \\ F &= \langle \boldsymbol{X}^{\mathfrak{m}}_{\xi}, \boldsymbol{X}^{\mathfrak{m}}_{\eta} \rangle, \\ G &= \langle \boldsymbol{X}^{\mathfrak{m}}_{\eta}, \boldsymbol{X}^{\mathfrak{m}}_{\eta} \rangle, \end{aligned} \tag{37}$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product in $\mathbb{R}^3$ restricted to Ω.

The most suitable measure of manifold curvature, used in our algorithm for adapting the geometry of the mesh approximating $\mathcal{M}_t$, is given by the *total curvature* (at $\mathfrak{m} \in \mathcal{M}_t$),

$$\kappa_T = \sqrt{\kappa_a^2 + \kappa_b^2} = \sqrt{2(2\mu^2 - K)}, \tag{38}$$

where κ_a and κ_b are the principal curvatures at the considered of point and μ is the *mean curvature* given by

$$\mu = \frac{1}{2}(\kappa_a + \kappa_b) = \frac{EN + GL - 2FM}{2(EG - F^2)}. \tag{39}$$

The Gaussian curvature, K, present in (38) is given by

$$K = \kappa_a \kappa_b = \frac{LN - M^2}{J^2}, \tag{40}$$

and the scalars L, M, N, appearing in (39) and (40), represent coefficients of the second fundamental form (see, for example, [28, §16.6]) given by

$$\begin{aligned} L &= \boldsymbol{n}^{\mathfrak{m}} \cdot \boldsymbol{X}^{\mathfrak{m}}_{\xi\xi}, \\ M &= \boldsymbol{n}^{\mathfrak{m}} \cdot \boldsymbol{X}^{\mathfrak{m}}_{\xi\eta}, \\ N &= \boldsymbol{n}^{\mathfrak{m}} \cdot \boldsymbol{X}^{\mathfrak{m}}_{\eta\eta}. \end{aligned} \tag{41}$$

When triangulating invariant manifolds with boundary (see Definition 2.3) it is very important to take into account the *geodesic curvature* of the boundary $\partial\mathcal{M}_t$, in addition to the curvatures of $\mathcal{M}_t$.

Given a surface curve, $\boldsymbol{\gamma}(s) : I\!R \hookrightarrow \mathcal{M}_{t_n}$, embedded in the manifold snapshot $\mathcal{M}_t \subset \Omega$, and a point $\mathfrak{m} = \boldsymbol{\gamma}(s^*) \in \mathcal{M}_{t_n}$, the geodesic curvature κ_g at $\mathfrak{m}$ is given by

$$\kappa_g = \kappa_\gamma \cos\psi. \tag{42}$$

The angle ψ in (42) is the angle between the surface normal $\boldsymbol{n}$ and the curve binormal $\boldsymbol{b}_\gamma$ (see figure 3) at the point $\mathfrak{m}$, and κ_γ is the curvature of $\boldsymbol{\gamma}$, given by

$$\kappa_\gamma = \frac{|\dot{\boldsymbol{\gamma}} \times \ddot{\boldsymbol{\gamma}}|}{|\dot{\boldsymbol{\gamma}}|^3}. \tag{43}$$

We will be interested in a particular curve embedded in $\mathcal{M}_t$ which corresponds to the boundary $\partial\mathcal{M}_t$ of the manifold snapshot.

4.2 Parameterisation of manifold snapshots $\mathcal{M}_t$

In order to compute curvatures and surface normals at any point $\mathfrak{m}$ of a manifold snapshot $\mathcal{M}_{t_n}$ in the ordered sequence $\{\mathcal{M}_{t_n}\}_{n\in Z_N}$ using the formulae derived in the previous subsection, we need to be able to determine a suitable (at least C^2) parameterisation of a patch $\sigma \subset \mathcal{M}_{t_n}$ containing $\mathfrak{m}$. One way of parameterising the manifold snapshots can be derived directly from the method introduced in §3 (approximating the invariant manifold, $\mathcal{M} \hookrightarrow \Omega \times I$, of the system (2) by the sequence $\{\mathcal{M}_{t_n}\}_{n\in Z_N}$). Such a procedure enables, in principle, a parameterisation of all subsequent manifold snapshots using coordinate charts defined on the initial snapshot, $\mathcal{M}_{t_0}$, and it is instructive to consider it first.

Let $(\sigma, \mathcal{L}_\sigma)$ be some coordinate chart on $\mathcal{M}_{t_0}$, so that $(\xi, \eta) \in \mathcal{L}_\sigma(\sigma)$ parameterises $\sigma \subset \mathcal{M}_{t_0}$ (cf. (34)). The patch σ can be mapped, for example, onto a patch $\tilde{\sigma} \subset \mathcal{M}_{t_n}$ with the help of (29) with ϕ_{t_n} given by (9), i.e.

$$\mathcal{M}_{t_n} \ni \tilde{\sigma} = \mathfrak{P}\big(\phi_{t_n}(\sigma, t_0)\big). \tag{44}$$

If now every point[6] $\mathfrak{m}$ in the patch σ is represented by $\boldsymbol{X}_{t_0}(\xi, \eta)$, for some $(\xi, \eta) \in \mathcal{L}(\sigma)$ and $\boldsymbol{X}_{t_0}$ given by (34), every point in the patch $\tilde{\sigma} \subset \mathcal{M}_{t_n}$ can be represented as

$$\boldsymbol{X}_{t_n}(\xi, \eta) = \mathfrak{P}\big(\phi_{t_n}(\boldsymbol{X}_{t_0}(\xi, \eta), t_0)\big). \tag{45}$$

Note in particular that if the initial manifold snapshot, $\mathcal{M}_{t_0}$, is a regular surface[7], all subsequent manifold snapshots in the sequence $\{\mathcal{M}_{t_n}\}_{n\in\mathbb{Z}^+}$ can be parameterised using the same global coordinate chart. Unfortunately, even if $\mathcal{M}_{t_0}$ is regular and such a global parameterisation exists, the mapping (45) is often very sensitive to numerical errors and renders numerical implementations of such a procedure unreliable. This is particularly the case when the manifold snapshots, given by the image of (45), are highly convoluted surfaces in $\Omega \subset I\!R^3$ - a situation which occurs generically when computing stable and unstable manifolds of hyperbolic trajectories over sufficiently long time intervals I.

We note here that there exist numerical methods for constructing global parameterisations of time-independent regular surfaces (e.g. [63, 45]), which in our setting would correspond to constructing a global parameterisation of $\mathcal{M}_{t_n}$ without using the mapping (45), and based solely on a discrete set of points contained in $\mathcal{M}_{t_n}$. However, despite some recent advances in this area (see [45]), parameterisation techniques based on such an approach do not perform well on highly convoluted surfaces which contain tightly packed folds.

An alternative strategy, which is algorithmical in nature and proves very useful in numerical applications, relies on deriving local parameterisations for a collection of neighbourhoods, $\{\sigma_j\}, \sigma_j \subset \mathcal{M}_{t_n}$, which contain the points of interest $\{\mathfrak{m}^j\}, \mathfrak{m}^j \in \mathcal{M}_{t_n}$. We outline the general idea below and describe details of a numerical implementation in §4.3.

[6] We will often refer to $\boldsymbol{X}(\xi, \eta)$ as a point on the manifold even though, strictly speaking it is an image of $(\xi^{\mathfrak{m}}, \eta^{\mathfrak{m}}) = \mathcal{L}(\mathfrak{m}) \in I\!R^2$ under the mapping $\mathcal{E} \circ \mathcal{L}^{-1}$, see (34) and (32).

[7] If a surface $S \hookrightarrow I\!R^n$ is regular, there exists a single coordinate chart (U, u) such that $S \hookrightarrow U \subset I\!R^n$ and $u : U \to I\!R^2$

Consider a point on the manifold snapshot $\mathfrak{m} \in \mathcal{M}_{t_n}$ and a suitably chosen set of three orthogonal, unit vectors $\{\mathfrak{e}_\xi, \mathfrak{e}_\eta, \mathfrak{e}_n\}$ (see §4.3) which can be used to define an alternative coordinate system on $\Omega \subset I\!R^3$ in the following way

$$\mathcal{A}_\mathfrak{m} : \ \Omega \to I\!R^3,$$

$$\begin{pmatrix} \xi \\ \eta \\ \nu \end{pmatrix} = \mathcal{A}_\mathfrak{m}(\boldsymbol{x}) \equiv \begin{pmatrix} \langle \boldsymbol{x} - \mathfrak{m}, \mathfrak{e}_\xi \rangle \\ \langle \boldsymbol{x} - \mathfrak{m}, \mathfrak{e}_\eta \rangle \\ \langle \boldsymbol{x} - \mathfrak{m}, \mathfrak{e}_n \rangle \end{pmatrix}, \tag{46}$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product in $I\!R^3$.

If the vectors $\{\mathfrak{e}_\xi, \mathfrak{e}_\eta, \mathfrak{e}_n\}$ are chosen in such a way that there exists a patch σ on the C^r manifold snapshot $\mathcal{M}_{t_n}$ containing $\mathfrak{m}$ (i.e. $\sigma \subset \mathcal{M}_{t_n}$, $\mathfrak{m} \in \sigma$), such that all points in σ are in the graph of some C^r function, $f : I\!R^2 \to I\!R$, given by

$$\text{graph } f = \{(\xi, \eta, \nu) \in \mathcal{A}_\mathfrak{m}(\sigma) \subset I\!R^3 : \ \nu = f(\xi, \eta)\}, \tag{47}$$

a local coordinate system on σ can be defined as

$$\mathcal{L}_\sigma : \ \mathcal{M}_{t_n} \supset \sigma \to I\!R^2,$$

$$(\xi, \eta, \nu(\xi, \eta)) \to (\xi, \eta). \tag{48}$$

The coordinates (ξ, η) can be thought of as parameterising a plane

$$\mathcal{T} = \{\boldsymbol{x} \in I\!R^3 : \quad (\boldsymbol{x} - \mathfrak{m}) \cdot \mathfrak{e}_n = 0\}, \tag{49}$$

in which case the last coordinate in (46) can be interpreted as the distance of a point $(\xi, \eta, \nu)^T$ from the plane $\mathcal{T}$.

Finally, using (34) and (48), the patch σ can parameterised in $I\!R^3$ as

$$\mathcal{E}(\sigma) \supset \boldsymbol{X}(\xi, \eta) = \mathcal{E} \circ \mathcal{A}_\mathfrak{m}^{-1} \circ \mathcal{L}_\sigma^{-1}(\xi, \eta) = \begin{pmatrix} \langle \mathfrak{m}, \boldsymbol{e}_x \rangle + \xi \langle \mathfrak{e}_\xi, \boldsymbol{e}_x \rangle + \eta \langle \mathfrak{e}_\eta, \boldsymbol{e}_x \rangle + f(\xi, \eta) \langle \mathfrak{e}_n, \boldsymbol{e}_x \rangle \\ \langle \mathfrak{m}, \boldsymbol{e}_y \rangle + \xi \langle \mathfrak{e}_\xi, \boldsymbol{e}_y \rangle + \eta \langle \mathfrak{e}_\eta, \boldsymbol{e}_y \rangle + f(\xi, \eta) \langle \mathfrak{e}_n, \boldsymbol{e}_y \rangle \\ \langle \mathfrak{m}, \boldsymbol{e}_z \rangle + \xi \langle \mathfrak{e}_\xi, \boldsymbol{e}_z \rangle + \eta \langle \mathfrak{e}_\eta, \boldsymbol{e}_z \rangle + f(\xi, \eta) \langle \mathfrak{e}_n, \boldsymbol{e}_z \rangle \end{pmatrix}. \tag{50}$$

Clearly, provided that the graph (47) can be constructed, the local parameterisation (50) allows for computing the desired local surface properties for every point $\mathfrak{m} \in \mathcal{M}_{t_n}$ in the patch σ using the formulae derived in §4.1. We show next how this procedure can be implemented numerically.

4.3 Numerical estimation of local manifold properties from discrete manifold data

As already described in §2.3, each snapshot $\mathcal{M}_{t_n}$ of the invariant manifold $\mathcal{M}$ is represented in our algorithm by the mesh $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$ which provides a piecewise-linear approximation of the manifold snapshot. Therefore, curvatures and normal vectors at points $\mathfrak{m} \in \mathcal{M}_{t_n}$ cannot be computed directly from the mesh, since the determination of these local properties requires at least a locally C^2 surface (cf. §4.1). Similar problems are common in computer graphics and surface reconstruction applications, and there exists a number of approaches for estimating curvatures and surface normals from polygonal (i.e. C^1) surfaces (see, for example, [27, 37, 40, 53]). These techniques can be roughly divided into two classes: 1) methods based on *discrete approximation formulas*, which attempt to estimate curvatures and normal vectors directly from the polygonal mesh, and 2) algorithms estimating local properties at points $\mathfrak{m} \in \mathcal{M}_{t_n}$ based on a *locally* $C^r (r \geqslant 2)$ *approximations* of suitably chosen patches $\sigma \subset \mathcal{M}_{t_n}$ such that $\mathfrak{m} \in \sigma$.

Probably the most prominent methods employing discrete approximation formulas are based on the Gauss-Bonnet theorem [15, 68]. However, although these methods are generally fast, we found them not suitable for the present applications due to their relatively low accuracy and poor performance on meshes corresponding to under-resolved approximations of the manifold snapshot $\mathcal{M}_{t_n}$. It is critical for our applications that the method used for the estimation of local manifold properties is robust in the sense that it does not lead to spurious results on

poor-quality meshes. In contrast to 'static' meshes, for example those reconstructing surfaces from densely sampled 3D laser scans, the first approximation of $\mathcal{M}_{t_{n+1}}$ can be only obtained from the current mesh, $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$, by advecting it with the help of $\phi_{t_{n+1}}$ (cf. (29)). Due to the fact that, generally, the geometry of $\mathcal{M}_{t_{n+1}}$ differs significantly from that of $\mathcal{M}_{t_n}$, and because $\phi_{t_{n+1}}$ does not preserve the optimal distribution of mesh vertices, the 'advected' mesh may yield locally insufficient resolution of $\mathcal{M}_{t_{n+1}}$.

In order to estimate local properties of each manifold snapshot in our algorithm, we use a method employing local analytical approximations of suitably chosen neighbourhoods, $\sigma \subset \mathcal{M}_{t_n}$, which are derived in a parametric form via a least-squares fit of a polynomial surface to the mesh vertices contained in σ in the coordinates (48). These methods are generally reliable in terms of stability and accuracy across a wide spectrum of cases, including non-uniform meshes (see [47] and references therein). The most important steps of such a procedure are listed below.

Local approximation of a manifold neighbourhood containing the vertex $\mathfrak{m}^i \in \mathcal{M}_{t_n}$:

(1) For each vertex $\mathfrak{m}^i \in \mathcal{M}_{t_n}$, determine a 'sufficient' number of its neighbours, using the first K neighbour rings, i.e. $\mathfrak{N}^i_K = \{\cup_{k=1}^{K}{}_k\mathfrak{N}^i_{t_n}\}$ (see (25), (27)).

(2) Construct the reference plane (49) and the set of orthogonal basis vectors $(\mathfrak{e}_\xi, \mathfrak{e}_\eta, \mathfrak{e}_n)$ associated with this plane.

(3) Decompose the neighbours $\mathfrak{N}^i_K$ in the coordinates (46).

(4) Construct a local approximation of the neighbourhood $\sigma \subset \mathcal{M}_{t_n}$, $\mathfrak{m}^i \in \sigma$, via the least squares fit of $\mathfrak{N}^i_K$ and $\mathfrak{m}^i$ to a polynomial surface (see (56) below) in the coordinates (48).

A few important comments are in order here:

Choosing the 'right' neighbours of the vertex $\mathfrak{m}^i \in \mathcal{M}_{t_n}$ (Step (1)). Note that the selection of neighbours of each $\mathfrak{m}^i \in \mathcal{M}_{t_n}$, based on its ring neighbours, prevents the algorithm from choosing mesh vertices which are only close to the vertex $\mathfrak{m}^i$ in the Euclidean metric on $\mathbb{R}^3$ and not in the (Riemannian) metric on the manifold. However, additional care is required when selecting the neighbours $\mathfrak{N}^i_n$, since the method proposed in §4.2 requires that the chosen vertices are all contained in the local graph (47) of a smooth function. Consequently, the parameterisation (48) can be constructed only if the vertices $\mathfrak{N}^i_n$ and the normal $\mathfrak{e}_n$, determining the reference plane $\mathcal{T}_\mathfrak{m}$ (49), are compatible in the sense that there exists a neighbourhood $\sigma \in \mathcal{M}_{t_n}$ such that $\mathfrak{m}^i, \mathfrak{N}^i_n \in \sigma$. Implementing this requirement in the algorithm is not straightforward and results in a trade-off between the accuracy of the approximation and the stability of the method. As long as the vertex $\mathfrak{m}^i$ and all its neighbours, $\mathfrak{N}^i_K$, are contained in a local graph (47) of some smooth function, the maximum order of the approximating polynomial surface, as well as the accuracy of a least-squares fit of this surface, increases with the number of selected neighbours. However, since the normal $\mathfrak{e}_n$ is determined in *Step (2)*, it is not known a priori which of the neighbours in $\mathfrak{N}^i_K$ can be contained in the graph (47) that contains $\mathfrak{m}^i$. If the chosen set of neighbours contains such 'wrong', vertices the subsequent least squares fit (*Step (4)*) can be seriously biased. In order to avoid such a situation, the steps (1-4) are repeated twice, and the second iteration involves additional filtering of the set of neighbours, $\mathfrak{N}^i_n$, of the examined vertex $\mathfrak{m}^i \in \mathcal{M}_{t_n}$. After the first run, the plane normal $\mathfrak{e}_n$ and the normals, $\mathfrak{n}^j$, to $\mathcal{M}_{t_n}$ at $\mathfrak{m}^j \in \mathfrak{N}^i_n$ are estimated. During the second run, the vertices in $\mathfrak{N}^i_n$ whose normals deviate from $\mathfrak{e}_n$ by more than some fixed angle are discarded, i.e. only neighbouring vertices which satisfy

$$\mathfrak{N}^i_{n\angle} = \{\mathfrak{m}^j \in \mathfrak{N}^i_n : \ |\langle \mathfrak{e}_n, \mathfrak{n}^j \rangle| \geqslant |\cos(\theta_{\mathrm{dev}})|\}, \quad 0 \leqslant \theta_{\mathrm{dev}} \leqslant \pi/2, \tag{51}$$

are accepted. Moreover, it is required that the image $\mathcal{L}_\sigma(\mathfrak{m}^i) \in \mathbb{R}^2$ is contained within a maximal polygon generated by the union of all possible polygons formed by the points $\mathcal{L}_\sigma(\mathfrak{N}^i_{n\angle}) \in \mathcal{L}_\sigma(\sigma) \subset \mathbb{R}^2$. This restriction provides also additional stability to the least-squares fit.

Determination of the reference plane (Step (2)). Determination of an appropriate reference plane $\mathcal{T}$, given by (49), is achieved by analysing the eigenvalues of a covariance matrix based on a spatial distribution of the neighbours $\mathfrak{N}^i_{n\angle}$. Similar techniques were used earlier

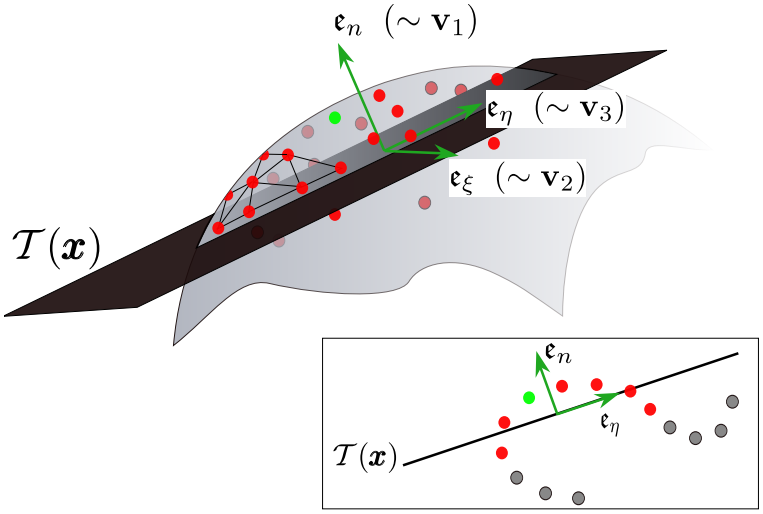

Figure 4: Schematic illustration of determination of the reference plane $\mathcal{T}(\boldsymbol{x})$ (see (54)) which is used to construct a local parameterisation (50) of a patch, $\sigma \subset \mathcal{M}_{t_n}$, containing the point $\mathfrak{m}^i$ (green) (see §4.3); the next stage of the parameterisation procedure is sketched in figure 5. The orientation of $\mathcal{T}(\boldsymbol{x})$, determined by $\mathfrak{e}_n$, is derived using the eigenvectors of the covariance matrix (52) of a collection of appropriately chosen points $\mathfrak{m}^j \in \mathfrak{N}^i_{n\angle} \subset \mathfrak{M}_{t_n}$ (red) which are neighbours of $\mathfrak{m}^i$ in the mesh.

in a different context (e.g. [58, 66]) in order to 'cluster' and simplify oversampled data sets obtained from 3D laser scans.

For a discrete set of J mesh vertices $\mathfrak{m}^j \in \mathfrak{N}^i_{K\angle} \subset I\!R^3$, the 3×3 covariance matrix has the following form

$$\boldsymbol{C} = \left[\mathfrak{m}^1 - \bar{\mathfrak{m}}, \mathfrak{m}^2 - \bar{\mathfrak{m}}, \ldots, \mathfrak{m}^J - \bar{\mathfrak{m}}\right] \cdot \left[\mathfrak{m}^1 - \bar{\mathfrak{m}}, \mathfrak{m}^2 - \bar{\mathfrak{m}}, \ldots, \mathfrak{m}^J - \bar{\mathfrak{m}}\right]^T, \tag{52}$$

where $\bar{\mathfrak{m}} = \left(\sum_{j=1}^J \mathfrak{m}^j\right)/J$ denotes the 'centre of gravity' of the neighbouring vertices $\mathfrak{N}^i_{n\angle}$. Since the covariance matrix is real and symmetric, its eigenvalues, $\lambda_{(1-3)}$, are all real-valued and its eigenvectors, $\mathbf{v}_{(1-3)}$, form an orthogonal coordinate frame. The eigenvalues measure the variation of the set $\mathfrak{N}^i_{n\angle}$ along the corresponding eigenvectors and the square of total variation from the center of gravity $\bar{\mathfrak{m}}$ is given by

$$\sum_j |\mathfrak{m}^j - \bar{\mathfrak{m}}|^2 = \lambda_1 + \lambda_2 + \lambda_3. \tag{53}$$

If the eigenvalues are ordered so that $\lambda_1 \leqslant \lambda_2 \leqslant \lambda_3$, the eigenvector $\mathbf{v_1}$ indicates the direction of the smallest 'spread' of the collection of vertices in $\mathfrak{N}^i_{K\angle}$. Consequently, the eigenvector $\mathbf{v_3}$ indicates the direction of the largest spread of the vertices $\mathfrak{m}^j$. It then follows that the plane,

$$\mathcal{T} = \{\boldsymbol{x} \in I\!R^3 : \ (\boldsymbol{x} - \bar{\mathfrak{m}}) \cdot \mathbf{v}_1 = 0\}, \tag{54}$$

minimizes the sum of squared distances to the vertices $\mathfrak{m}^j$ (see [66, 25]), i.e. it represents the least-squares plane through the set $\mathfrak{N}^i_{K\angle}$. The basis $(\mathfrak{e}_\xi, \mathfrak{e}_\eta, \mathfrak{e}_n)$, discussed in §4.2, is constructed in the following way: (1) The orientation of the plane normal, $\mathfrak{e}_n \sim \mathbf{v}_1$, is chosen in such a way that it maximizes the scalar product $\langle \mathfrak{e}_n, \mathfrak{n}^i \rangle$; (2) The remaining two basis vectors, $\mathfrak{e}_\xi \sim \mathbf{v}_2$ and $\mathfrak{e}_\eta \sim \mathbf{v_3}$, are chosen so that the resulting set of orthogonal vectors is right-handed (see figure 4).

Using the basis vectors $(\mathfrak{e}_\xi, \mathfrak{e}_\eta, \mathfrak{e}_n)$, it is possible represent the vertex $\mathfrak{m}^i \in \mathfrak{M}_{t_n}$ and its neighbours $\mathfrak{N}^i_{n\angle} \subset \mathfrak{M}_{t_n}$ as

$$\begin{pmatrix} \xi^i \\ \eta^i \\ \nu^i \end{pmatrix} = \mathcal{A}_{\bar{\mathfrak{m}}}(\mathfrak{m}^i), \quad \begin{pmatrix} \xi^j \\ \eta^j \\ \nu^j \end{pmatrix} = \mathcal{A}_{\bar{\mathfrak{m}}}(\mathfrak{N}^i_{K\angle}), \quad j \in \mathrm{Idx}[\mathfrak{N}^i_{K\angle}], \tag{55}$$

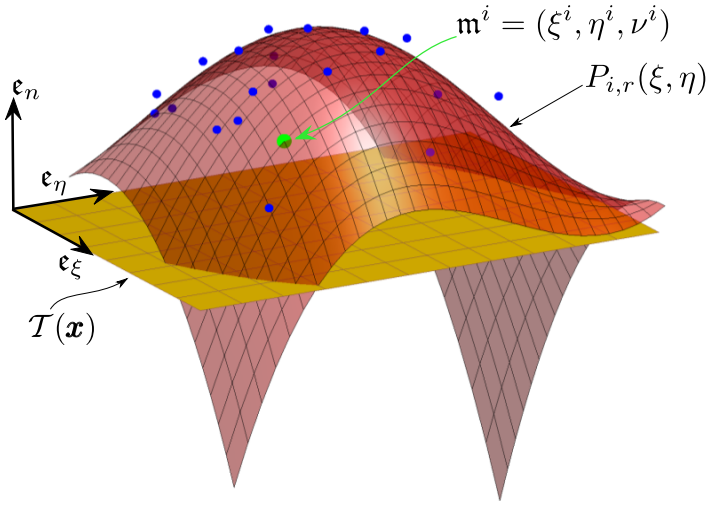

Figure 5: Construction of a local analytical approximation (red) of a patch σ of the manifold snapshot $\mathcal{M}_{t_n}$ from a discrete set of mesh vertices (mesh not shown). The patch σ contains the manifold point $\mathfrak{m}^i$ (green) at which the curvature and the normal vector are to be estimated. A polynomial surface (56) is fitted in the least-squares sense to the collection of neighbours of $\mathfrak{m}^i$, given by $\mathfrak{N}^i_{n\angle}$ (51), which are decomposed in the coordinate frame $(\mathfrak{e}_\xi, \mathfrak{e}_\eta, \mathfrak{e}_n)$ using (55).

where $\mathcal{A}_{\bar{\mathfrak{m}}}$ denotes the coordinates (46) which are determined with respect to the origin at $\bar{\mathfrak{m}}$. Provided that the vertices and the were chosen in an appropriate way (cf. discussion preceding (51)), the local coordinates parameterising the manifold patch $\sigma \subset \mathcal{M}_{t_n}$ can be obtained using (48).

Least-squares local surface approximation (Step (4)). The method used here for reconstructing the manifold patch, $\sigma \subset \mathcal{M}_{t_n}$, which contains the considered point $\mathfrak{m}^i \in \mathfrak{M}_{t_n}$, is based on an approximation of the graph (47) by a polynomial surface in the variables (48). The approximating polynomial surface is required to contain the point $\mathfrak{m}^i$ and minimize the distance to the neighbouring vertices $\mathfrak{N}^i_{K\angle}$ in the least-squares sense. We use this method here to estimate the local properties of $\mathcal{M}_{t_n}$ at $\mathfrak{m}^i$.

Assuming that the mesh vertex $\mathfrak{m}^i \in \mathfrak{M}_{t_n}$ and its neighbours $\mathfrak{N}^i_{K\angle}$ are appropriately chosen and decomposed in the coordinates $\mathcal{A}$ (cf. (55)), consider a surface $\mathfrak{S}_r = \big(\xi, \eta, P_r(\xi, \eta)\big)$ given by a graph of a bivariate polynomial of order n,

$$P_{i,r}(\xi, \eta) = \nu^i + \sum_{l=0}^{r} c_l (\xi - \xi^i)^l (\eta - \eta^i)^{r-l}. \tag{56}$$

Note that $\mathfrak{S}_r$ always passes through $\mathfrak{m}^i = (\xi^i, \eta^i, \nu^i)^T$ regardless of the values of the $r(r+3)/2$ coefficients c_l of the polynomial $P_{i,r}$.

The requirement that the polynomial surface $\mathfrak{S}_r$ approximates the graph (47) containing the neighbours $\mathfrak{N}^i_{K\angle}$ could be satisfied, in principle, by choosing $\tilde{K} = r(r+3)/2$ vertices from $\mathfrak{N}^i_{K\angle}$ and solving the following set of linear equations for the coefficients c_l:

$$P_{i,r}(\xi^k, \eta^k) = \nu^k, \qquad k = 1, 2, ..., r(r+3)/2. \tag{57}$$

In such a case, the approximating surface interpolates $\mathfrak{m}^i$ and its $\tilde{K}$ chosen neighbours and takes no account of the remaining ones. A different choice of $\tilde{K}$ neighbours from the set $\mathfrak{N}^i_{K\angle}$ will generally lead to different results. Consequently, such an approach often results in estimation of inconsistent values of curvature and normals of $\mathcal{M}_{t_n}$ at neighbouring mesh vertices. Moreover, the non-uniform distribution of the neighbours of $\mathfrak{m}^i$ within $\mathcal{L}_\sigma(\mathfrak{N}^i_{K\angle}) \in I\!R^2$ often leads to an ill-conditioned matrix of the system (57). This fact has been recognised earlier in algorithms developed in the context of surface reconstruction from 3D laser scans where the

surface points are acquired with experimental errors (see, for example, [20, 31, 38, 52]). In the present case, the measurement errors are replaced by numerical errors due to integration of the vertex trajectories and, more importantly, possible errors introduced during a mesh refinement process (cf. §5.1 and §5.2) due to interpolation of new points between existing mesh vertices. Moreover, the errors in determination of the coefficients c_l are amplified further, leading to a spurious oscillatory behaviour, when computing derivatives of high-order polynomial surfaces $\mathfrak{S}_r$. Therefore, in order to obtain consistent estimates of curvature and normals of $\mathcal{M}_{t_n}$, we search for a low-degree-polynomial approximation of the manifold, by solving a deliberately overdetermined system (57) in the least-squares sense (see figure 5). The most common method in this class uses the least-squares paraboloidal fitting, i.e. $\mathfrak{S}_2$ (see [53, 10, 18, 29]). However, we found that performing a least squares fit of a bi-quartic polynomial surface, $\mathfrak{S}_4$, yields much more accurate results, at least for the purpose of the current application. In order to determine coefficients of the bi-quartic polynomial, at least $K = 14$ neighbours are needed which usually requires collecting neighbours from the first three rings of $\mathfrak{m}^i$ (i.e. $\mathfrak{N}^i_{3\angle}$). The higher order ($r = 4$ in (56)) of the fitted polynomial surface can, in some cases, lead to inconsistent results when the distribution of the vertex neighbours is highly non-uniform. This possibility is monitored especially on the vertices contained in the boundary of $\mathcal{M}_{t_n}$ and, if necessary, a bi-cubic or the paraboloidal fitting is used instead.

When the local approximation has been determined, the patch $\sigma \subset \mathcal{M}_{t_n}$ can be (approximately) parameterised using (50), and the surface normal and curvature at $(\xi^i, \eta^i) = \mathcal{L}_\sigma(\mathfrak{m}^i)$ can be computed using (35) and (38).

4.4 Interpolation of manifold points from mesh vertices

The mesh adaptation algorithms discussed in §5, in particular the mesh refinement (cf. §5.1) and remeshing (cf. §5.3) procedures, require information about the local manifold properties at manifold points which are not necessarily contained within the existing mesh. This requires the ability to interpolate the manifold between the existing mesh vertices $\mathfrak{M}_{t_n}$ in order to either insert a new vertex into the mesh or to estimate manifold curvatures at some additional points. The local properties properties of $\mathcal{M}_{t_n}$, including the surface normals and curvatures at existing mesh vertices are known from the previous step and can be computed using the procedures outlined in the previous subsection. The first step of the interpolation process is very similar to the steps described earlier except that now we decompose all vertices in the neighbourhood within which the interpolation is needed with respect to the same reference plane.

The interpolation is performed using a procedure introduced in [21], referred to as the modified Shephard's method, which was proposed to overcome drawbacks of the original interpolation scheme introduced by Shephard [67]. Further extensions of this method, employing the radial basis functions to interpolate multivariate data sets, can be found in [42]. The modified Shephard's method has many advantages such as numerical efficiency, good reproduction quality, stability and inherent parallelism.

Assume we want to interpolate $\mathcal{M}_{t_n}$ at a point $\boldsymbol{p}$, given K neighbouring[8] vertices $\mathfrak{m}^1_p, \ldots, \mathfrak{m}^K_p$ in the approximating mesh $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$. The local coordinates for a manifold patch containing the points $\mathfrak{m}^k_p$ are constructed in a way analogous to that described in §4.3. Consequently, we denote the coordinates of $\mathfrak{m}^k_p$ by $(\xi^k, \eta^k) = \mathcal{L}(\mathfrak{m}^k)$ (cf. (48)), and the corresponding points in a local graph approximating the manifold by (ξ^k, η^k, ν^k). Following [21], the interpolating function, defined in the local variables (48) on a manifold patch $\sigma \subset \mathcal{M}_{t_n}$ has the following form:

$$F(\xi, \eta) = \sum_{k=1}^{K} W_k(\xi, \eta) Q_k(\xi, \eta), \tag{58}$$

where the *nodal functions*, Q_k, satisfy $Q_k(\xi^k, \eta^k) = \nu^k$ and fit the values of the remaining points in the least-squares sense. Here, we simply use the nth order bivariate polynomials, described in the previous subsection, as the nodal functions, i.e. $Q_k(\xi, \eta) = P_{k,n}(\xi, \eta)$. The only difference between the polynomials used here for interpolation and previously for curvature

[8] Determination of neighbouring points for a point not contained within the mesh is not a trivial task, especially when the manifold consists of densely packed folds. The selection process is generally based on identifying relevant ring neighbours and is described in relevant subsections; see, in particular §2.3, §4.3 and §5.

and surface normal estimation is that the nodal functions associated with each vertex involved are computed in the same local coordinates.

The weight functions $\bar{W}_k$ are defined as

$$\bar{W}_k = \frac{W_k(\xi,\eta)}{\sum_{k=1}^{K} W_k(\xi,\eta)}\,, \tag{59}$$

where, denoting $r_k = \sqrt{(\xi-\xi^k)^2+(\eta-\eta^k)^2}$,

$$W_k(\xi,\eta) = \left[\frac{(r_{W_k}-r_k)_+}{r_{W_k} r_k}\right]^{\alpha}, \quad \alpha > 2, \tag{60}$$

and

$$(r_{W_k}-r_k)_+ = \begin{cases} r_{W_k}-r_k, & \text{if } r_k < r_{W_k}, \\ 0, & \text{if } r_k \geqslant r_{W_k}. \end{cases} \tag{61}$$

The 'radius of influence' about a point $(\xi^k,\eta^k) = \mathcal{L}_\sigma(\mathfrak{m}^k)$, denoted by r_{w_k} in (60) and (61), is chosen large enough to include the neighbours $(\xi^k,\eta^k) = \mathcal{L}_\sigma(\mathfrak{N}^i_{n\angle})$. As a consequence the value ν^k at the point $\mathcal{L}(\mathfrak{m}^k) = (\xi^k,\eta^k)$ influences interpolated values at points within the distance r_{W_k}. We chose this radius to include all of the neighbours $\mathfrak{m}^k$.

The weight functions (59) satisfy the cardinality relations, i.e.

$$\bar{W}_k(\xi^j,\eta^j) = \delta_{jk} \quad j,k \in 1,2,\ldots,K, \tag{62}$$

and they constitute a partition of unity [9] since they are also normalised to satisfy

$$\sum_{k=1}^{K} \bar{W}_k(\xi,\eta) = 1. \tag{63}$$

Note finally, choosing the weight functions (60) with $\alpha > 2$ implies that, apart from being continuous within $\mathcal{L}_\sigma(\sigma) \subset I\!R^2$, at least the first two partial derivatives of (60) are zero at the nodal points $(\xi^k,\eta^k) = \mathcal{L}_\sigma(\mathfrak{m}^k_p)$. This implies that the interpolating function is C^2 within $\sigma \subset \mathcal{M}_{t_n}$ and it inherits the local manifold properties at the vertices from the respective nodal functions Q_k.

5 Mesh adaptation

The triangular meshes, $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$, approximating the manifold $\mathcal{M}$ in the algorithm at the series of observational times $\{t_n\}_{n\in Z_N}$, are generated as a trade-off between the accuracy of the approximation of each $\mathcal{M}_{t_n}$, the computational time required for the triangulation, and the available storage capacity. An optimal way to reconcile these constraints is to generate meshes which, for each manifold snapshot $\mathcal{M}_{t_n}$, contain smaller triangles in areas where the manifold geometry changes rapidly (i.e. areas of high curvature) while covering the 'flat' areas of $\mathcal{M}_{t_n}$ with larger triangles. Moreover, it is desirable that a mesh which is 'adapted' to the manifold geometry also contains triangles of high quality (i.e. triangles close to equilateral) since such a representation provides the best accuracy of the mesh approximation (see, for example, [3] for more details). A high quality mesh provides good conditioning of the local-surface approximation or interpolation procedures (see §4.3 and §4.4) which reduces the possibility of appearance of spurious oscillations in the curvature data.

In this section we describe methods which are implemented in our algorithm for adjusting the mesh resolution (i.e. *mesh refinement*, §5.1) and for improving the mesh quality (i.e. *mesh regularization*, §5.2). These algorithms are used after every advection step from t_n to the next observation time t_{n+1} (see steps 5 and 8 in figure 2).

5.1 Mesh refinement

Assume that the mesh $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$ approximating the manifold snapshot $\mathcal{M}_{t_n}$ is such that all triangles in the mesh are nearly equilateral, and such that the density of the mesh vertices $\mathfrak{M}_{t_n}$, distributed throughout $\mathcal{M}_{t_n}$, is higher in regions of high curvature on the manifold

[9] See, for example, [11, p. 88].

snapshot. If this mesh is then advected to the next observation time t_{n+1} using (29) (see also §3), the mesh triangles formed by the advected mesh vertices $\mathfrak{M}_{t_{n+1}}$ are likely to be significantly distorted from their counterparts at t_n. Moreover, if the geometry of $\mathcal{M}_{t_n}$ and $\mathcal{M}_{t_{n+1}}$ differs significantly, the distribution of the advected mesh vertices $\mathfrak{M}_{t_{n+1}}$ within $\mathcal{M}_{t_{n+1}}$ does not coincide with regions of high curvature on $\mathcal{M}_{t_{n+1}}$. Therefore, unless the invariant manifold is time-independent (i.e. $\mathcal{M}_{t_n} = \mathcal{M}_{t_{n+1}}$, see Definition 2.1), it is necessary to monitor and automatically refine the subsequent meshes in order to maintain desired resolution of high-curvature regions on the manifold snapshots.

A mesh whose vertices were advected from t_n to the next observation time t_{n+1} using (29) is represented by $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_n})$; the connectivity data remains unaffected by the advection (cf. §3). The required resolution of the advected manifold is maintained by comparing the length of each edge in the advected mesh, i.e. $\Delta X^{ij} = |\mathfrak{m}^i_{t_{n+1}} - \mathfrak{m}^j_{t_{n+1}}|$, $\mathbb{M}^{t_n}_{ij} \neq 0$, against a target edge length

$$\Delta X^{ij}_{target} = 1/\big(R\, f(\kappa(\mathfrak{m}^i, \mathfrak{m}^j))\big), \tag{64}$$

where the parameter R controls the global mesh resolution and the function f expresses dependence of ΔX^{ij}_{target} on local properties of the manifold at the edge vertices, $\kappa(\mathfrak{m}^i, \mathfrak{m}^j)$. Different choices of the function f and κ are possible. If, for example, $f(\kappa) = (\kappa_T(\mathfrak{m}^i) + \kappa_T(\mathfrak{m}^j))/2$, where the total curvature κ_T is given by (38), the scaling (64) keeps the same number of triangles on a sphere regardless of its radius (see [6]). Here, in order to take account of the possible existence of the boundary $\partial\mathcal{M}_{t_n}$, we choose

$$\kappa(\mathfrak{m}^i, \mathfrak{m}^j) = \begin{cases} (\kappa_T(\mathfrak{m}^i) + \kappa_T(\mathfrak{m}^j))/2 + \beta(g(\mathfrak{m}^i) + g(\mathfrak{m}^j))/2, & \text{if} \quad \mathfrak{m}^i, \mathfrak{m}^j \in \mathfrak{B}_{t_n}, \\ (\kappa_T(\mathfrak{m}^i) + \kappa_T(\mathfrak{m}^j))/2 + \beta g(\mathfrak{m}^i), & \text{if} \quad \mathfrak{m}^i \in \mathfrak{B}_{t_n}, \mathfrak{m}^j \notin \mathfrak{B}_{t_n}, \\ (\kappa_T(\mathfrak{m}^i) + \kappa_T(\mathfrak{m}^j))/2, & \text{if} \quad \mathfrak{m}^i, \mathfrak{m}^j \notin \mathfrak{B}_{t_n}, \end{cases} \tag{65}$$

where the discretised boundary $\mathfrak{B}_{t_n}$ of $\mathcal{M}_{t_n}$ is given by (24); κ_T and g are, respectively, the total curvature (38) and the geodesic curvature (42) at the edge vertices $\mathfrak{m}^i, \mathfrak{m}^j$. The parameter $\beta \geqslant 0$ in (65) determines the importance of the boundary resolution in the computations.

We now choose $f(\kappa)$ in such a way that it prevents generation of excessively large or excessively small triangles. Moreover, it is important from the point of view of stability of the interpolating algorithms (cf. §4.4) and the subsequent remeshing (cf. §5.3) that the triangle edge length changes gradually even across regions on the manifold characterised by rapid curvature changes. In order to achieve the above requirements, we set

$$f(\kappa) = (\kappa_{max} - \kappa_{min}) \left[\frac{\kappa}{1+\kappa}\right]^{\alpha} + \kappa_{min}, \quad \alpha > 1, \tag{66}$$

where α in (66) is a smoothing parameter, and κ_{max} and κ_{min} denote, respectively, the maximum and the minimum curvature cut-off.

Using (64) together with (66), every edge in the advected mesh $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}_{t_n})$ is examined and mesh edges which do not satisfy the target length criterion are identified (Step 4 in figure 2). If necessary, a new vertex is introduced at the midpoint of a pre-image of each long edge in the mesh at the previous time step, i.e. $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$. Location of the new vertex on the manifold snapshot $\mathcal{M}_{t_n}$ is then determined by an interpolation in local coordinates which are constructed for a manifold patch containing suitable neighbours of the two vertices located at the ends of the long edge. The interpolation procedure is analogous to that described in §4.3 and §4.4. Additional connections are added to the connectivity matrix in such a way that the new vertex is common to the four newly created triangles (see Step 5 in figure 2 and figure 6c). The new vertices are subsequently advected to t_{n+1} and the manifold curvature is computed at all vertices of the refined mesh $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}})$. The edge lengths are then re-examined using the updated curvature data and the advected mesh is refined further, if necessary, in an analogous way by inserting vertices into the mesh at $t = t_n$ and advecting them to $t = t_{n+1}$. This process is terminated when no edges in the advected and refined mesh $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}})$ exceed the target length (64).

5.2 Mesh regularization

The refinement procedure performed on the advected mesh in the way described above often generates poor quality triangles which, if not properly handled, may lead to spurious numerical

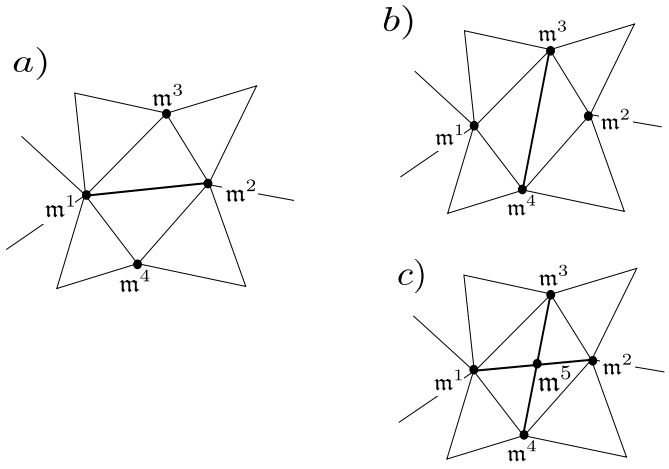

Figure 6: Illustration of the basic steps of mesh adaptation discussed in §5. $a) \rightarrow b)$ edge swap attempting to reduce mesh irregularity (67); see §5.2 for a discussion. $a) \rightarrow c)$ mesh edge refinement (cf. §5.1) performed when the edge length exceeds the local target edge length ΔX^{12}_{target}, given by (64).

errors in the estimation of curvatures and normal vectors at points of $\mathcal{M}_{t_{n+1}}$. However, a refined mesh can be improved by changing the connectivity of the mesh vertices in a process referred to as *mesh relaxation*, and by moving the vertices on $\mathcal{M}_{t_{n+1}}$ - a procedure referred to as *mesh smoothing*. The relevant methods implemented in the algorithm are briefly described below. Significant mesh improvement can be achieved after only few passes of these algorithms which, for best results, are applied to each vertex in the refined mesh $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}})$ in a random order.

5.2.1 Connectivity regularization (mesh relaxation)

In their most general formulation, algorithms of this type perform a series of local operations that modify the vertex connectivity, $\mathbb{M}^{t_n}$ (cf. (23)), in the refined mesh $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_n})$ in order to minimize *mesh irregularity* subject to a geometrical constraint that the altered mesh yields a valid triangulation in $I\!R^3$ (i.e. the triangular faces do not intersect). The mesh irregularity is defined as

$$\mathcal{C}(\mathbb{M}^{t_n}) = \sum_j (d(\mathbb{M}^{t_n}_{j,:}) - \mathfrak{d}^j_{opt})^2, \tag{67}$$

where $d(\mathbb{M}^{t_n}_{j,:})$ is the number of non-zero entries in the j-th row of the connectivity matrix and, as discussed in §2.3, it indicates the number of its first-ring neighbours, ${}_1\mathfrak{R}^i_{t_n}$, to which a mesh vertex $\mathfrak{m}^j \in \mathfrak{M}_{t_{n+1}}$ is directly connected. Hereafter, we refer to this quantity as the *degree of a vertex* $\mathfrak{m}^j$ and denote it by $\mathfrak{d}^j$, i.e.

$$\mathfrak{d}^j \equiv d(\mathbb{M}^{t_n}_{j,:}). \tag{68}$$

The parameters $\mathfrak{d}^j_{opt}$ in (67) denote an *optimal degree* of each mesh vertex which, for triangular meshes, is taken to be $\mathfrak{d}_{opt} = 4$ for boundary vertices, and $\mathfrak{d}_{opt} = 6$ for interior vertices. Such an optimisation problem is nontrivial due the fact that the mesh irregularity (67) has, in general, many local minima. Attempts to converge to a global minimum by employing stochastic methods in order to escape from the local minima result in a significant computational load and slow convergence (see [2]). Here, we follow a simplified approach, similar to that proposed in [22, 2], which attempts to minimise the mesh irregularity (67) in a series of uncorrelated basic *edge flips* in order to minimise the *degree residual*, given by $(\mathfrak{d}^j - \mathfrak{d}^j_{opt})$, of each mesh vertex (see figure 7).

The geometry considered for each edge of the mesh $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}})$ is sketched in figure 6. Here we focus on the edge $\overline{\mathfrak{m}^1\mathfrak{m}^2}$ and consider whether swapping it for $\overline{\mathfrak{m}^3\mathfrak{m}^4}$ is beneficial for lowering the degree residuals of the vertices in the quadrilateral formed by the two triangles adjacent to $\overline{\mathfrak{m}^1\mathfrak{m}^2}$. The reduction of mesh irregularity obtained by such a swap can be measured

in terms of a *relaxation index*[10] which is defined, for a given configuration, as a difference of the degree residuals of vertices in the quadrilateral $\Diamond\mathfrak{m}^1\mathfrak{m}^2\mathfrak{m}^3\mathfrak{m}^4$ contained in the edge and those that are not. Consequently, the relaxation index for the configuration of figure 6a is given by

$$\begin{aligned} E_{12\_34} &= (\mathfrak{d}^1 - \mathfrak{d}^1_{opt}) + (\mathfrak{d}^2 - \mathfrak{d}^2_{opt}) - (\mathfrak{d}^3 - \mathfrak{d}^3_{opt}) - (\mathfrak{d}^4 - \mathfrak{d}^4_{opt}) \\ &= \mathfrak{d}^1 + \mathfrak{d}^2 - (\mathfrak{d}_3 + \mathfrak{d}^4) + \mathfrak{E}_{12\_34}, \end{aligned} \tag{69}$$

where $\mathfrak{d}^j = d(\mathbb{M}^{t_n}_{j,:})$ is the degree of each vertex and $\mathfrak{d}^j_{opt}$ is its optimal degree and

$$\mathfrak{E}_{12\_34} = \mathfrak{d}^3_{opt} + \mathfrak{d}^4_{opt} - (\mathfrak{d}^1_{opt} + \mathfrak{d}^2_{opt}). \tag{70}$$

Since the swap $\overline{\mathfrak{m}^1\mathfrak{m}^2} \rightarrow \overline{\mathfrak{m}^3\mathfrak{m}^4}$ decreases $\mathfrak{d}_1$ and $\mathfrak{d}_2$ and increases $\mathfrak{d}_3$ and $\mathfrak{d}_4$, the relaxation index for the alternative configuration, E_{34_12} (figure 6b), can be expressed as

$$E_{34\_12} = (\mathfrak{d}^3 + 1) + (\mathfrak{d}^4 + 1) - \big(\mathfrak{d}^1 + 1\big) - (\mathfrak{d}^2 + 1) - \mathfrak{E}_{12\_34} = 4 - E_{12\_34}. \tag{71}$$

Clearly, it is desirable to swap the edge $\overline{\mathfrak{m}^1\mathfrak{m}^2}$ for $\overline{\mathfrak{m}^3\mathfrak{m}^4}$ only when $E_{34\_12} > E_{12\_34}$. Thus, based on (71), the swap is desirable if and only if $E_{12\_34} > 2$. In the *neutral case*, when $E_{12\_34} = E_{34\_12} = 2$, the desirability of the swap $\overline{\mathfrak{m}^1\mathfrak{m}^2} \rightarrow \overline{\mathfrak{m}^3\mathfrak{m}^4}$ is the same as that of swapping it back and no action is taken.

Since the considered quadrilateral $\Diamond\mathfrak{m}^1\mathfrak{m}^2\mathfrak{m}^3\mathfrak{m}^4$ is contained in the mesh which is embedded in $\Omega \subset I\!R^3$ and approximates $\mathcal{M}_{t_{n+1}}$, two geometrical constraints have to be satisfied (besides the relaxation index condition $E_{12\_34} > 2$) in order that the edge swap is feasible. The first requirement is that the two new faces, $f_I = \Delta\{\mathfrak{m}^1, \mathfrak{m}^3, \mathfrak{m}^4\}$ and $f_{II} = \Delta\{\mathfrak{m}^2, \mathfrak{m}^3, \mathfrak{m}^4\}$, generated by the edge swap do not intersect any other faces in the mesh. If the faces do not intersect, two error measures, introduced in [69], are used to control the fidelity of the new mesh based on the original mesh[11]. The first measure, $\mathfrak{E}_{smth}$, indicates how well the given face coincides with tangent planes estimated individually from its vertices, and is given by

$$\mathfrak{E}_{smth}(f) = \max_{j\in\{1,2,3\}} \langle \boldsymbol{n}_f, \boldsymbol{n}^j \rangle, \tag{72}$$

where $\boldsymbol{n}_f$ is the face normal and $\boldsymbol{n}^j$ is a vector normal to $\mathcal{M}_{t_{n+1}}$ at the jth vertex given by $\mathfrak{m}^j$ (cf. (35)). If either $\mathfrak{E}_{smth}(f_I) > \cos(\theta_{smth})$ or $\mathfrak{E}_{smth}(f_{II}) > \cos(\theta_{smth})$, where θ_{smth} is some user-specified threshold angle, the edge swap is not performed.

The second measure, given by

$$\mathfrak{E}_{dist}(f) = \max_{i,j\in\{1,2,3\}} \langle \boldsymbol{n}^i, \boldsymbol{n}^j \rangle, \tag{73}$$

provides a rough indication of average curvature of a patch of $\mathcal{M}_{t_{n+1}}$ containing the face vertices. The larger the variation between the curvatures at the face vertices, the larger the distance between the triangular face and the true manifold snapshot is likely to be. Therefore, the edge swap is not performed if either $\mathfrak{E}_{dist}(f_I) > \cos(\theta_{smth})$ or $\mathfrak{E}_{dist}(f_{II}) > \cos(\theta_{smth})$, where θ_{dist} is some user-specified threshold angle.

The algorithm begins by performing geometrically feasible swaps of edges $\overline{\mathfrak{m}^i\mathfrak{m}^j}$ for which $E_{ij\_kl} \geqslant 4$, updating the connectivity data structure and the edge relaxation index data structure after every edge swap (Only the relevant entries in these structures are modified). Ideally, this stage of the relaxation terminates if there are no edges in the mesh with the relaxation index $E_{ij\_kl} \geqslant 4$, except for the edges unavailable for swapping due to the geometrical constraints. It can be easily shown that such a process can be achieved in a finite number of operations (see [22]). However, for large meshes the number of needed operations can be huge and, in practice, the computational cost of completing the relaxation process far exceeds the gains due to improved mesh regularity. Therefore, we constrain the relaxation process by not allowing any edge to be swapped more than twice. The relaxation process is then performed for edges with $E_{ij\_kl} \geqslant 3$ in an analogous way.

We note that it is often beneficial to perform a swap the neutral edges (i.e. $E_{ij\_kl} = 2$) followed by another $E_{ij\_kl} \geqslant 4$ and $E_{ij\_kl} \geqslant 3$ relaxations. This is due to the fact that, although

[10] See [22] for more details regarding this definition.

[11] Ideally, one should verify the fidelity of the new mesh approximation based on the manifold snapshot $\mathcal{M}_{t_{n+1}}$ which is generally unavailable a priori in our applications.

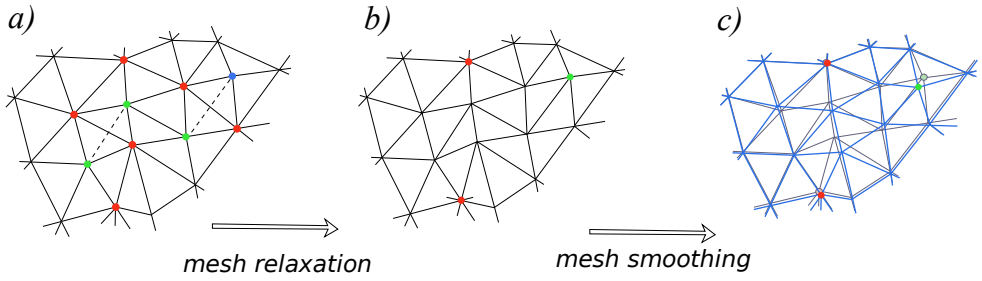

Figure 7: Schematic illustration of mesh regularization techniques which are performed on the refined mesh in order to improve its quality. Mesh relaxation (a,b), performed globally using only the connectivity data $\mathbb{M}^{t_{n+1}}$, attempts to 'push' the number of edges connected at each node towards 6. Mesh smoothing is performed on mesh patches in local coordinates and attempts to make the triangles equilateral.

it does not reduce the mesh irregularity, swapping neutral edges can 'shift' the mesh faults to regions where they provide additional flexibility to the mesh. Such a procedure can eventually lead to a global mesh relaxation (cf. [22]) but it is usually time consuming and we do not apply it in the algorithm.

5.2.2 Vertex relocation (mesh smoothing)

The mesh smoothing procedure attempts to improve the mesh quality by shifting the mesh vertices (ideally, along the manifold snapshot) so that the mesh triangles are close to equilateral. Consider a vertex $\mathfrak{m}^i \in \mathfrak{M}_{t_{n+1}}$ and a patch $\sigma_\Delta \subset (\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}})$ in the refined, relaxed triangular mesh approximating the manifold snapshot $\mathcal{M}_{t_{n+1}}$; σ_Δ is formed by the union of all triangles in the mesh which have $\mathfrak{m}^i$ as their vertex. The mesh vertices located on the boundary of σ_Δ are the first ring neighbours of $\mathfrak{m}^i$ contained in ${}_1\mathfrak{R}^i_{t_n}$ (see (25)). After the advection of the mesh (to t_{n+1}) and a subsequent mesh refinement (§5.1), the mesh patch σ_Δ is likely to contain poor quality triangles. We want to find a new location of $\mathfrak{m}^i$ within σ_Δ, denoted by $\tilde{\mathfrak{m}}^i \in \mathcal{M}_{t_{n+1}}$, such that the angles of all the triangles in the mesh patch are maximized. A solution to this problem directly in $\mathbb{R}^3$ involves generally a computationally expensive optimisation problem, which may be non-convex or have non-linear constraints. Instead, we solve this problem approximately in local coordinates in $\mathbb{R}^2$, which parameterise a manifold patch σ containing $\mathfrak{m}^i$ and its first ring neighbours, and then project the result onto $\mathcal{M}_{t_{n+1}}$.

Assume that $\mathfrak{m}^i$ has K first ring neighbours which lie on the boundary of the mesh patch σ_Δ. These boundary vertices can be represented in conveniently chosen local coordinates in $\mathbb{R}^2$, i.e. based on (48) and and the discussion in §4.3 we write $\mathfrak{p}^k \equiv (\xi^k, \eta^k) = \mathcal{L}_\sigma(\mathfrak{m}^k),\ k = 1, \ldots, K$, and $\mathfrak{c}^i \equiv (\xi^i, \eta^i) = \mathcal{L}_\sigma(\mathfrak{m}^i)$. The new location of the interior vertex, $\tilde{\mathfrak{c}}^i$, is obtained using a hybrid scheme which employs *Laplacian smoothing* [9] and *weighted angle-based smoothing* [70]. The Laplacian smoothing, which attempts to place $\tilde{\mathfrak{c}}^i$ at the 'centre of mass' of its neighbours $\bar{\mathfrak{p}} = \sum_{k=1}^K \mathfrak{p}^k/K$, is by far the most efficient representative of smoothing methods and it is applied first. However, this procedure can generate invalid (inverted) faces in mesh regions with highly irregular connectivity (see below) by placing $\tilde{\mathfrak{c}}^i$ outside the boundary of the projected patch $\mathcal{L}_\sigma(\sigma_\Delta)$. If such a scenario is detected, the weighted angle-based smoothing is applied in an attempt to equalise each pair of adjacent, interior edges in the patch σ_Δ. If we denote by α_k the angle between the edges $\overline{\mathfrak{p}^{k-1}\mathfrak{p}^k}$ and $\overline{\mathfrak{p}^k\mathfrak{p}^{k+1}}$ the optimal location of the new interior vertex is given by

$$\tilde{\mathfrak{c}}^i = \left(\sum_{k=1}^{K} \frac{\mathfrak{c}^k_\angle}{\alpha_k^2} \right) \Big/ \sum_{k=1}^{N} \frac{1}{\alpha_k^2} \tag{74}$$

where $\mathfrak{c}^k_\angle$ is a point located on the bisector of α_k such that $|\mathfrak{p}^k - \mathfrak{c}^k_\angle| = |\mathfrak{p}^k - \mathfrak{c}^i|$. The location of the new interior vertex, $\tilde{\mathfrak{m}}^i \in \mathcal{M}_{t_{n+1}}$, is obtained by inverse-mapping $\tilde{\mathfrak{c}}^i$ (i.e. by computing $\mathcal{L}_\sigma^{-1}(\tilde{\mathfrak{c}}^i)$) in the way identical to that described in §4.4.

Generally, the above hybrid method is highly successful in improving the mesh triangle quality. However, in some extreme cases, when the boundary of the projected patch $\mathcal{L}_\sigma(\sigma_\Delta)$

is far from convex, the method may still fail to produce a valid triangulation of the patch σ_Δ. In such cases the location of the interior vertex $\mathfrak{m}^i$ is not updated and the algorithm moves to the next vertex in the mesh $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}})$.

We remark that there exist other methods for improving mesh quality which have not been implemented yet in our algorithm. In particular, mesh relaxation methods involving area-based smoothing and subsequent Delaunay edge-flip (e.g. [69] and references therein) definitely deserve attention in further developments.

5.3 Manifold remeshing

The procedures for refining and improving the advected mesh described above can successfully control the mesh fidelity and adapt it in regions where the evolving manifold requires higher resolution. However, while some regions of the manifold become increasingly convoluted in time, other regions no longer require the resolution they needed at earlier stages of the evolution. In order to optimise the computational cost of our automated algorithm, we also need to remove vertices from the regions on the manifold which no longer require high concentration of small triangles. Unfortunately, the removal of vertices by applying an 'inverse' algorithm to that outlined in §5.3 is not straightforward. Removing vertices by simply collapsing long edges in the mesh, based on the local target edge length, can result in introducing 'faults' into the mesh which cannot (at least at this stage of the development) be later removed using the methods discussed in §5.2. This subsequently leads to large numerical errors in the algorithms estimating the local properties of the manifold snapshots (§4). We note here that there exist a wealth of methods that can cope with the task of vertex removal in applications related to surface reconstruction from range data in computer graphics (see, for example, [58, 59, 66, 24, 25, 1]). In particular the method presented in [69], combining the mesh relaxation techniques with vertex relocation and removal, is fast and yields robust results. However, the successive mesh alterations and vertex relocation used in these methods lead to an accumulation of interpolation errors if the vertex density on the surface is low. In particular, the insertion of new vertices via local surface interpolation, using neighbouring vertices which were inserted at an intermediate stage of the same remeshing process, may lead to spurious errors. This is generally not a serious problem in the case of algorithms reconstructing (time-independent) boundaries of a 3D object from oversampled range data which allows robust vertex clustering and a more accurate interpolation of new vertices. However, when applying similar techniques to simplify the evolving manifold meshes which, for reasons of computational efficiency maintain the minimum required vertex density, the accumulation of the interpolation errors cannot be handled easily and it is often further amplified due to the manifold evolution.

Therefore, we find it beneficial for the purpose of our applications to generate, if needed, an entirely new mesh with desired properties. This action is taken if the number of edges which are shorter than the desired local target edge length exceed certain proportion of all mesh edges; the typical threshold value used is 60%. The remeshing process is performed by applying the *advancing front technique* to obtain a regular triangulation which conforms to externally specified criteria, such as the local triangle size and boundary constraints. The new mesh generation still involves vertex interpolation but, since the triangulation process is well structured, the number of such operations is kept to a minimum.

The advancing front method was initially developed for triangulating planar domains (e.g. [9, 46, 26, 65]) and later extended to triangulate regular surfaces in three-dimensions (e.g. [56, 34, 33, 61]). Here we use a variant of the advancing front technique which generates the triangular mesh directly in $\mathbb{R}^3$ using the refined but irregular mesh as a reference in order to estimate the manifold geometry. The main modification introduced here is necessitated by the need to triangulate surfaces which often contain tightly packed folds. The main obstacle to using the existing point cloud triangulation algorithms is that they operate using the standard Euclidean metric in $\mathbb{R}^3$ and cannot distinguish between points lying close to each other in the sense of Riemannian distance on the surface and points which are just close to each other in $\mathbb{R}^3$. This problem is remedied here by using the connectivity information of the reference mesh, $\mathbb{M}^{t_{n+1}}$, in order to estimate the (Riemannian) distances on the manifold snapshot.

Implementation of the advancing front method for direct surface triangulation

When triangulating a planar domain, the method requires as an input a closed oriented boundary which is subsequently discretised, according to the user-specified criteria, yielding the

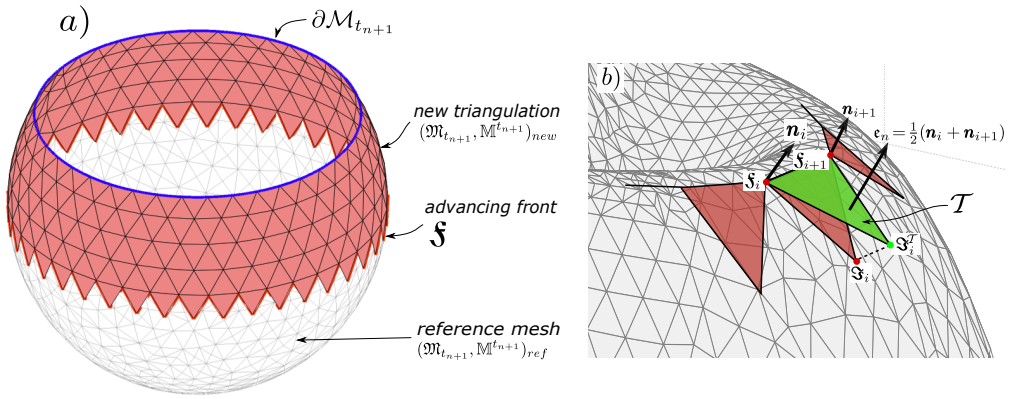

Figure 8: Schematic illustration of remeshing of a manifold snapshot $\mathcal{M}_{t_{n+1}}$ discussed in §5.3. The remeshing is performed directly in $I\!R^3$ using the refined but not optimal mesh $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}})$. The *ideal* vertex $\mathfrak{F}_i$ is associated with each new segment, $\overline{\mathfrak{F}_i\mathfrak{F}_{i+1}}$, of the advancing front $\mathfrak{F}$, by first considering the local geometry projected onto the reference plane $\mathcal{T}$ (cf. (49)), determined from the average of the normal vectors $\boldsymbol{n}_i$ and $\boldsymbol{n}_{i+1}$ (i.e. $\mathfrak{e}_n^i = \frac{1}{2}(\boldsymbol{n}_i + \boldsymbol{n}_{i+1})$. The projection $\mathcal{T} \ni \mathfrak{F}_i^{\mathcal{T}} \rightarrow \mathfrak{F}_i \in \mathcal{M}_{t_{n+1}}$ is computed in the way discussed earlier in §4.3 (see also figure 5).

initial *front* composed of an ordered set of nodes. Each segment of the front (between the nodes) is then treated as a side of a new triangle and a location of the third vertex, which generates a new triangle, is then determined. A number of factors are involved in the determination of the third vertex, making up the new triangle. The most important geometrical constraints are that the new vertex does not lie inside an existing triangle and that the two new edges are within the local target-length tolerance margins. The new vertex location is deemed *ideal* if the triangle created in the process is equilateral. The front is updated after every addition of a triangle to the mesh so that it remains on the closed boundary of the triangulated region. Finally, the front closes on itself which marks the end of the triangulation process.

A similar procedure applied to triangulating a surface in three-dimensions is inevitably more complicated but can be performed efficiently in most cases, provided that the mesh triangle size is not allowed to change sharply across the surface. We note first that a rather straightforward extension of the technique to three-dimensions is possible when the surface to be triangulated is regular (i.e. can be globally parameterised using a single coordinate chart). However, even if such a parameterisation is possible, the triangulation algorithm often suffers severely from amplification of numerical errors, which arises due to repeated transformations between the 2D parameter space, where the triangulation is performed, back to $I\!R^3$. This effect is particularly pronounced in regions where the coordinate chart has large derivatives and was found to be a very common problem when computing the evolution of stable and unstable manifolds of DHTs, often leading to invalid meshes in $I\!R^3$.

We therefore apply the advancing front method directly in $I\!R^3$ using the refined, irregular mesh as a reference for estimation local manifold properties and interpolation. The main steps of the algorithm remeshing a manifold snapshot with boundary are outlined below. The algorithm remeshing manifold snapshots given by closed surfaces executes the same steps, except that two fronts are advanced on both sides of the initial front given by the closed curve $\mathfrak{L}_{t_{n+1}}$ (see §3.1).

(1) Determine the initial front $\mathfrak{F}$ by discretising the boundary $\partial\mathcal{M}_{t_{n+1}}$ according to the required resolution parameter R.

Discretisation of the boundary, $\partial\mathcal{M}_{t_{n+1}}$, which serves as the initial front for the new triangulation of $\mathcal{M}_{t_{n+1}}$, is a very important step which largely determines the success of the whole procedure. The discretisation procedure interpolates new boundary vertices between vertices of a reference boundary which can, in principle, be extracted from the refined, irregular reference mesh $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}})_{old}$. However, in order to improve the accuracy of this process, the high-resolution copy of the boundary, $\mathfrak{B}_{t_{n+1}}$ (cf. §3.1), is used instead of $\mathfrak{B}_{t_{n+1}}$ (cf. (24)). Recall that $\mathfrak{B}_{t_{n+1}}$ is computed independently from the mesh $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}})$ using the algorithm discussed in Appendix B. The separation between the new boundary vertices is determined by the target edge length (64) which, for boundary vertices, takes into account the geodesic curvature g (see (42) and figure 3) of the boundary in addition to the manifold curvature κ_T (see (38)). The boundary points of the new mesh are simply chosen from the vertices $\mathfrak{B}_{t_{n+1}}$ by marching along the (closed)

boundary and selecting vertices whose separation does not exceed the local target edge length $\Delta X_{target}(\kappa_T, g)$, given by (64). Each vertex $\mathfrak{b}^i \in \mathfrak{B}_{t_{n+1}}$ in the boundary contains information about the local curvature κ_γ at $\mathfrak{b}^i$ (see (43)), as well as indices of two nearest vertices, say $\mathfrak{m}^1$ and $\mathfrak{m}^2$, of the reference mesh. Hereafter, we will refer to these two nearest vertices in the reference mesh as the *parents* of $\mathfrak{b}^i$. Similar, data structure is later constructed and maintained for every other point in the new mesh which allows for selecting the right vertices in the reference mesh at the curvature estimation and interpolation stages (cf. §4.3 and §4.4). The curvatures g and κ_T at $\mathfrak{b}^i$ are estimated using κ_γ and an average of normal vectors of the first-ring neighbours (see (25)) of $\mathfrak{m}^1$ and $\mathfrak{m}^2$.

(2) Compute the location of an *ideal* vertex, $\mathfrak{V}_i$, for each segment of the front $\overline{\mathfrak{F}_i\mathfrak{F}_{i+1}}$. This process is initiated in the reference plane $\mathcal{T}$ (cf. (49)) which contains the two *base vertices*, $\mathfrak{F}_i$ and $\mathfrak{F}_{i+1}$, and is perpendicular to

$$\mathfrak{e}_n = \tfrac{1}{2}(\boldsymbol{n}_i + \boldsymbol{n}_{i+1}), \tag{75}$$

where $\boldsymbol{n}_i$ and $\boldsymbol{n}_{i+1}$ are vectors normal to $\mathcal{M}_{t_{n+1}}$ at the base vertices (see figure 8b). The location of the ideal vertex in $\mathcal{T}$ is given by

$$\mathfrak{V}_i^{\mathcal{T}} = \frac{1}{2}(\mathfrak{F}_i + \mathfrak{F}_{i+1}) + \frac{\sqrt{3}}{2}\frac{\boldsymbol{t}_i}{|\boldsymbol{t}_i|}\Delta X_{target}^{i,i+1}, \tag{76}$$

where $\Delta X_{target}^{i,i+1}$ is given by (64) and $\boldsymbol{t}_i$ is a vector contained in $\mathcal{T}$ and orthogonal to the base segment $\overline{\mathfrak{F}_i\mathfrak{F}_{i+1}}$, i.e.

$$\boldsymbol{t}_i = \mathfrak{e}_n \wedge (\mathfrak{F}_{i+1} - \mathfrak{F}_i). \tag{77}$$

The height of the triangle $\Delta\mathfrak{F}_i\,\mathfrak{V}_i^{\mathcal{T}}\,\mathfrak{F}_{i+1}$ is chosen in such a way that the two edges formed by connecting the base to $\mathfrak{V}_i^{\mathcal{T}}$, i.e. $\overline{\mathfrak{F}_i\mathfrak{V}_i^{\mathcal{T}}}$ and $\overline{\mathfrak{F}_{i+1}\mathfrak{V}_i^{\mathcal{T}}}$, are adjusted to the target edge length $\Delta X_{target}^{i,i+1}$. We relax here the traditional notion of the ideal vertex, and only require that connecting the base $\overline{\mathfrak{F}_i\mathfrak{F}_{i+1}}$ to $\mathfrak{V}_i^{\mathcal{T}}$ generates an isosceles triangle, rather than an equilateral one. The curvature value used to evaluate $\Delta X_{target}^{i,i+1}$ is based on an average curvature determined from reference mesh vertices which are sufficiently close to the base vertices, $\mathfrak{F}_i$ and $\mathfrak{F}_{i+1}$, in the Riemannian metric on $\mathcal{M}_{t_{n+1}}$. The appropriate reference mesh vertices, say $\mathfrak{N}$, are selected by identifying the first-ring neighbours of the parents (see **(1)**) of the two base vertices.

The ideal vertex $\mathfrak{V}_i$ is computed by projecting $\mathfrak{V}_i^{\mathcal{T}}$ onto $\mathcal{M}_{t_{n+1}}$ using (50). The curvature and normal vector at $\mathfrak{V}_i$ are estimated in the same way as described in §4.3. The parents of the ideal vertex are taken to be the two vertices contained in $\mathfrak{N}$ which are closest to it in the standard Euclidean metric on $\mathbb{R}^3$.

In order to improve the accuracy of this step, the final location of the ideal vertex is determined by repeating the above procedure (i.e. eqns. (75)-(77)), where $\mathfrak{e}_n$, $\boldsymbol{t}_i$ and ΔX_{target} are based on the averaged values of the curvatures and normals at the three vertices $\mathfrak{F}_i$, $\mathfrak{F}_{i+1}$, $\mathfrak{V}_i$.

(3) Choose a front segment, $\overline{\mathfrak{F}_i\mathfrak{F}_{i+1}}$, and identify a set of front vertices, $\mathfrak{f} \subset \mathfrak{F}$, which represent suitable candidates for the third vertex $\mathcal{V}$ in a new triangle $\Delta\mathfrak{F}_i\mathcal{V}\mathfrak{F}_{i+1}$.

Choose a segment on the front and find neighbouring front vertices contained in a ball of a radius $r \propto \Delta X_{target}$ (cf. (64)) centered at the mid-base point. The search is performed first in $\mathbb{R}^3$ and the ball radius is usually chosen as $r = 1.5\,\Delta X_{target}$. If there are any front vertices in the ball, their parents (see **(1)**) are examined in order to make sure that the selected front vertices are indeed close (in the Riemannian metric on $\mathcal{M}_{t_{n+1}}$) to the parents of the ideal vertex. The identified vertices serve as alternative candidates for the third vertex, in addition to $\mathfrak{V}_i$.

(4) Select the third vertex, $\mathcal{V}$. The 'front candidates', $\mathfrak{f}$ (see **(3)**), are examined first in order to check whether a triangle formed with one of them is of sufficiently good quality (see (78) below). It is generally beneficial to connect to an existing front vertex even at the expense of a reduction of the triangle quality. Insertion of an ideal vertex close to existing front vertices often leads to generation of poor quality triangles at later stages of the triangulation.

The selection of the third vertex, complementing the two base vertices, involves several steps performed in the local coordinates which parameterise a manifold patch containing the base vertices. It constitutes the most computationally intensive part of the triangulation process. The main steps are:

1) *Intersection test for the ideal vertex.* Check if a triangle formed by connecting the ideal vertex to the base overlaps with any of the existing triangles.
2) *Angle test of the ideal vortex.* Check the angles that edges of each prospective triangle would form with the front segments adjacent to the base $\overline{\mathfrak{S}_i\mathfrak{S}_{i+1}}$, i.e. the angles $\widehat{\mathfrak{V}\mathfrak{S}_i\mathfrak{S}_{i-1}}$ and $\widehat{\mathfrak{V}\mathfrak{S}_{i+1}\mathfrak{S}_{i+2}}$. If any of the angles falls below a threshold (usually $\alpha_{thr} = 40^\circ$), discard the ideal vertex.
3) *Front vertex ordering.* Order the front candidates for the third vertex, $\mathfrak{F}$, identified in **(3)**, with decreasing quality of triangles which would be generated by connecting them to the base $\overline{\mathfrak{S}_i\mathfrak{S}_{i+1}}$.
4) *Intersection test for the front candidate.* Check if the first front vertex in the sequence determined in the previous step (i.e. the front vertex $\mathfrak{F}_1$ which forms the best triangle with the base) intersects any existing triangles. If it does not, choose it as the *front candidate*. If it does, discard the vertex (i.e. $\mathfrak{F}_{new} = \mathfrak{F}_{old}\backslash\mathfrak{F}_1$) and examine the next front vertex.
5) *Angle test for the front candidate.* Check the angles that edges of each prospective triangle would form with the front segments adjacent to the base, i.e. the angles $\widehat{\mathfrak{F}_1\mathfrak{S}_i\mathfrak{S}_{i-1}}$ and $\widehat{\mathfrak{F}_1\mathfrak{S}_{i+1}\mathfrak{S}_{i+2}}$. If any of the angles falls below a threshold (usually $\alpha_{thr} = 40^\circ$), discard the vertex.

Let Q_{ABC} denote the quality of a triangle $\triangle ABC$ defined as

$$Q_{ABC} = \frac{\frac{1}{2}|\overrightarrow{AB} \times \overrightarrow{AC}|}{\overline{AB}^2 + \overline{BC}^2 + \overline{CA}^2}. \tag{78}$$

Provided that the ideal vertex passed the intersection and angle tests (1-2), it is chosen as the third vertex if:

i) $\mathfrak{F} = \emptyset$.

ii) $\mathfrak{F} \neq \emptyset$ but $Q_{\mathfrak{S}_i\mathfrak{S}_{i+1}\mathfrak{F}_1} < \delta_{thr} Q_{\mathfrak{S}_i\mathfrak{S}_{i+1}\mathfrak{V}}$;

iii) $\mathfrak{F} \neq \emptyset$, $Q_{\mathfrak{S}_i\mathfrak{S}_{i+1}\mathfrak{F}_1} \geqslant \delta_{thr} Q_{\mathfrak{S}_i\mathfrak{S}_{i+1}\mathfrak{V}}$ but

$$min\big(\widehat{\mathfrak{F}_1\mathfrak{S}_i\mathfrak{S}_{i-1}}, \widehat{\mathfrak{F}_1\mathfrak{S}_{i+1}\mathfrak{S}_{i+2}}\big) < min\big(\widehat{\mathfrak{V}\mathfrak{S}_i\mathfrak{S}_{i-1}}, \widehat{\mathfrak{V}\mathfrak{S}_{i+1}\mathfrak{S}_{i+2}}\big).$$

If the ideal vertex did not pass the intersection test (1) and $\mathfrak{F} \neq \emptyset$, then $\mathfrak{F}_1$ is chosen as the third vertex.

If the ideal vertex failed (2) but not (1), the angle test is also relaxed on the front vertices and the ideal vertex is chosen if any of (i-iii) holds.

Otherwise, when there is no suitable candidates, one of the front vertices adjacent to the base is relocated, i.e. the adjacent front edge, $\overline{\mathfrak{S}_i\mathfrak{S}_{i-1}}$ or $\overline{\mathfrak{S}_{i+1}\mathfrak{S}_{i+2}}$, which forms a smaller angle with the base $\overline{\mathfrak{S}_i\mathfrak{S}_{i+1}}$ is relocated along, respectively, a line $\overline{\mathfrak{V}\mathfrak{S}}_{i-1}$ or $\overline{\mathfrak{V}\mathfrak{S}}_{i+2}$ until the intersection test is satisfied.

(5) Add the new triangle $\Delta\mathfrak{S}_i\mathcal{V}\mathfrak{S}_{i+1}$ to the mesh by updating $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}})_{new}$.
Update the list of vertex locations $\mathfrak{M}_{t_{n+1}}$ and the connectivity data $\mathbb{M}^{t_{n+1}}$. Determine the manifold curvature κ_T and the normal at the new vertex $\mathcal{V}$. This is accomplished using an analogous procedure to that described in §4.3. Assuming that the two nearest neighbours of $\mathcal{V}$ in in the old mesh are $\mathfrak{m}^i$ and $\mathfrak{m}^j$, the list of neighbours of $\mathcal{V}$ used in the procedure is taken to be $\mathfrak{N}^{\mathcal{V}} = \mathfrak{N}^i_K \cup \mathfrak{N}^j_K$ (cf. Step (1) in §4.3).

(6) Update the front $\mathfrak{S}$.
If the vertex selected in **(4)** was the ideal one, the base segment is removed from the front $\mathfrak{S}$ and the two new edges are added. If the selected front vertex was a neighbour of one

of the base vertices, the base segment is swapped with the appropriate new edge. If the selected front vertex is not a neighbour, the front is split into two subfronts, one of which is stored in a *front repository* for a later triangulation.

If the front closed on itself as a result of the front update, another front is retrieved from the repository. If the repository is empty, the triangulation process is finished and the steps (8), (9) are skipped.

(7) Find ideal vertices for the new front segments generated in (6). This step is analogous to **(2)**.

(8) Go to **(3)**.

After completing the triangulation procedure, the generated mesh is regularized using a few passes of the regularizing algorithms (cf. §5.2 and Stage 8 in figure 2). The need for regularization arises from the fact that the mesh generated in regions where the front began closing on itself is often not ideal and may still contain poor quality triangles which were generated when there were no acceptable ideal vertices. Another common situation when non-ideal triangles are created occurs when the required mesh resolution is too low to accurately resolve the geometry of $\mathcal{M}_{t_{n+1}}$.

6 Performance and preliminary applications

We illustrate here how the developed algorithm (currently written in MATLAB) can be used to compute codimension one stable and unstable manifolds of hyperbolic trajectories in three-dimensional unsteady fluid flows. Computation of such manifolds is the most important application of the developed techniques from our point of view. Details concerning determination of the Distinguished Hyperbolic Trajectories (DHT) in the 3D non-autonomous setting, whose manifolds we compute here, are discussed in Appendix A. As pointed out in the preceding discussion, similar procedure can be applied to study the structure of other invariant manifolds in three-dimensional fluid flows. For example, the algorithm can be used to compute the evolving geometry of material surfaces in 3D fluid flows (both steady and unsteady). We postpone a detailed discussion of various applications and accuracy of the developed techniques, which are not confined to fluid-dynamical framework, to a subsequent publication.

6.1 Computation of stable and unstable manifolds of hyperbolic trajectories in unsteady three dimensional flows

In order to illustrate the performance of the algorithm in computations of stable and unstable manifolds of hyperbolic trajectories[12], we consider velocity fields obtained by perturbing the well known steady solution of equations of an inviscid incompressible fluid flow given by the Hill's spherical vortex (see, for example, [4]). The Hill's vortex flow, $\boldsymbol{H}$, is then perturbed by a time-dependent strain, $\boldsymbol{S}$, in the following way

$$\boldsymbol{v} = \boldsymbol{H}(x, y, z) + \boldsymbol{S}(x, y, z, t). \tag{79}$$

The components of the steady Hill's vortex in Cartesian coordinates are

$$\left.\begin{aligned} H_x &= (u_r \sin\Theta + u_\Theta \cos\Theta)\cos\Phi, \\ H_y &= (u_r \sin\Theta + u_\Theta \cos\Theta)\sin\Phi, \\ H_z &= (u_r \cos\Theta - u_\Theta \sin\Theta), \end{aligned}\right\} \tag{80}$$

where $r = \sqrt{x^2+y^2+z^2}$, $\Theta = \mathrm{acos(z/r)}$, $\Phi = \mathrm{acos}(x/\sqrt{x^2+y^2})$ and, assuming that a denotes the radius of the vortex, the velocity components in the spherical coordinates are

$$u_r = \begin{cases} U(1-a^3/r^3)\cos\Theta & \text{if} \quad r \geqslant a, \\ -\frac{3}{2}U(1-r^2/a^2)\cos\Theta & \text{if} \quad r < a, \end{cases} \tag{81}$$

[12]The algorithm to compute hyperbolic trajectories in the 3D setting, including a special class of *Distinguished Hyperbolic Trajectories (DHT's)* is described in Appendix A

$$u_\Theta = \begin{cases} -U(1 + a^3/(2r^3)) \sin\Theta & \text{if} \quad r \geqslant a, \\ \frac{3}{2} U(1 - 2r^2/a^2) \sin\Theta; & \text{if} \quad r < a. \end{cases} \tag{82}$$

This unperturbed (steady) Hill's vortex flow has two hyperbolic stagnation points

$$h_1 = (0, 0, -a)^T, \quad h_2 = (0, 0, a)^T, \tag{83}$$

which are located on the (flow-invariant) axis of symmetry $\boldsymbol{e}_z$ of the vortex.

The perturbing, time-dependent straining flow is given by

$$\boldsymbol{S} = \mathcal{A}(t) \cdot \mathcal{E}^{-1} \cdot \begin{bmatrix} \alpha(t) & 0 & 0 \\ 0 & \beta(t) & 0 \\ 0 & 0 & \gamma(t) \end{bmatrix} \cdot \mathcal{E} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{84}$$

where $\mathcal{A}(t)$ is a time-dependent strain amplitude, the strain rates are normalised so that $\max(\alpha, \beta, \gamma) = 1$ and they satisfy $\alpha + \beta + \gamma = 0$. The orthogonal matrix $\mathcal{E}$ is given by

$$\mathcal{E} = \begin{bmatrix} \cos\psi\cos\varphi - \sin\psi\cos\theta\sin\varphi & \cos\psi\sin\varphi + \sin\psi\cos\theta\cos\varphi & \sin\psi\sin\theta \\ -\sin\psi\cos\varphi - \cos\psi\cos\theta\sin\varphi & -\sin\psi\sin\varphi + \cos\psi\cos\theta\cos\varphi & \cos\psi\sin\theta \\ \sin\theta\sin\varphi & -\sin\theta\cos\varphi & \cos\theta \end{bmatrix}. \tag{85}$$

The Euler angles (θ, φ, ψ) used to parameterise $\mathcal{E}$ are defined in a standard way and are, in general, functions of time.

A flow of this type was considered earlier in an analytical work on the stability of the Hill's vortex to an axisymmetric, irrotational perturbation [54] and two follow-up publications on the vortex response to a three-dimensional perturbation [23, 60]. In these works, the authors assumed that diffusion was negligible and they focused on an evolution of a material surface (i.e. invariant manifold in the flow) bounding the rotational core of the perturbed vortex. This surface was shown to develop a spiky structure which was swept downstream as the flow evolved. It is of particular interest from the point of view of the Lagrangian methods developed here that, as the authors point out in [54], both entrainment and detrainment processes associated with perturbations of the Hill's vortex flow can be understood within the context of purely inviscid analysis.

Our 'dynamical systems' approach to the problem is purely kinematic in the sense that the time dependence in the flow is imposed externally. The dynamical system considered is associated with the system of equations

$$\dot{\boldsymbol{x}} = \boldsymbol{H}(\boldsymbol{x}) + \boldsymbol{S}(\boldsymbol{x}, t), \quad \boldsymbol{x} \in I\!R^3, \tag{86}$$

where $\boldsymbol{H}$ and $\boldsymbol{S}$ are given by, respectively, (80) and (84).

In what follows, we compute the stable and unstable manifolds of the so-called Distinguished Hyperbolic Trajectories [32] in different flow configurations. The DHT's are computed using an algorithm described in Appendix A. In all examples discussed below, we set $\alpha = \beta = -0.5$ which leads to an axisymmetric straining flow (in strain coordinates) whose axis of symmetry, corresponding to the strain rate γ, is inclined to $\boldsymbol{e}_z$ at the angle $\theta(t)$ (see figure 9b). The flow (79) can be made either axisymmetric or fully three-dimensional by an appropriate choice of the functions $\big(\theta(t), \varphi(t), \psi(t)\big)$. We will exploit both these possibilities in the following sections.

We note that, although we only use the velocity field (79) in order to illustrate the performance of our algorithm, similar analysis can be performed for dynamically consistent (or experimentally measured) flows. Such a study, which we postpone to a future publication, has the potential to provide a valuable insight into transport properties in complicated time-dependent, three-dimensional flows; in particular, the mixing processes taking place in the neighbourhood of three-dimensional vortices.

6.1.1 Comparison with two-dimensional results

Consider first an axisymmetric flow obtained from (79) by setting $\theta = 0$ so that the axis of symmetry of the strain and the vortex are aligned, and every plane containing $\boldsymbol{e}_z$ is invariant. In such

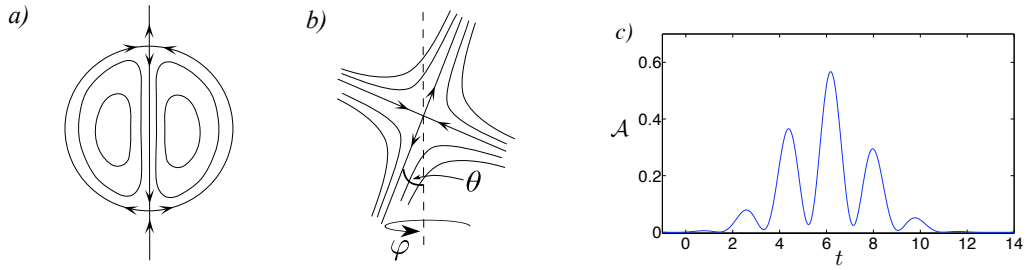

Figure 9: Schematic representation of a three-dimensional flow used in computations of invariant manifolds in §6. The steady Hill's spherical vortex (a), sketched in a symmetry plane, is perturbed by a time-dependent strain (b) whose axis of symmetry is inclined to the axis of symmetry of the Hill's vortex, $\boldsymbol{e}_z$, by an angle $\theta(t)$. The rotation of the strain axis with respect to $\boldsymbol{e}_z$ is determined by $\varphi(t)$. The amplitude of the strain changes with time as shown in c).

a case, it is possible to compare the results obtained from applying our 3D algorithm for computing invariant manifolds with results obtained from two-dimensional computations performed in one of the invariant planes. The strain amplitude $\mathcal{A}(t) = 0.3\sin(2t)\exp(-(t-6)^2/3.5^2)$ is chosen for computations in this section.

The resulting axisymmetric flow has two hyperbolic instantaneous stagnation points[13] (ISPs) throughout the evolution which are located on the axis of symmetry, $\boldsymbol{e}_z$, of the original Hill's vortex. For a small amplitude, $\mathcal{A} \ll 1$, of the perturbing strain, $\boldsymbol{S}$, these ISPs remain near the stagnation points, h_1 and h_2 (cf. (83)), of the unperturbed Hill's vortex. We denote them as ISP_1 and ISP_2. The 'smallness' of the perturbing amplitude is not a necessary condition for a successful computation of the DHTs and their manifolds but it provides a manifold configuration which is easier to understand in these preliminary examples. There are two DHTs in the flow which can be computed with the help of the algorithm described in Appendix A, using the paths traced by the ISPs as the initial guesses. We denote these DHTs by $\boldsymbol{\gamma_1}$ and $\boldsymbol{\gamma}_2$, respectively, according to the ISPs used to compute them. The trajectory $\boldsymbol{\gamma}_1$ has a three-dimensional unstable manifold (in the extended phase space) and the trajectory $\boldsymbol{\gamma}_2$ has a three-dimensional stable manifold (in the extended phase space). We stress here that the use of ISPs for the DHT computations is just a particular choice of the 'initial guess' for the iterative 'DHT-finding' algorithm. In fact, other suitably chosen C^1, *frozen-time hyperbolic* curves in the extended phase manifold can be used for this purpose (see Appendix A and Definition A.3 for more details). As discussed in §2.1, the manifold snapshots representing the instantaneous geometry of these manifolds are given by 2D surfaces embedded in $\mathbb{R}^3$. The computation of a compact region of the unstable manifold is initiated by placing a small disc (i.e. manifold with boundary; cf. Definition 2.3), centred around $\boldsymbol{\gamma}_1$ at some initial time t_0 ($t_0 = 0$ in this case), in such a way that it lies in the unstable subspace of the linearisation of the flow around $\boldsymbol{\gamma}_1(t_0)$. This initial disc is then triangulated using the methods described in §5.3 and 'fed into' the main algorithm, as indicated in figure 2.

Figure 10 shows an example of evolving geometry of a compact region (isotopic to a disc) of the unstable manifold of $\boldsymbol{\gamma}_1$ obtained from the 3D algorithm (red surface). The 3D computations were performed with $R = 5, \kappa_{min} = 3, \kappa_{max} = 150$. The resulting mesh contains approximately 7.4×10^4 vertices and over 15×10^4 triangles (approx. 3.7×10^4 vertices and 7.5×10^4 triangles used in the section shown in figure 10). The relatively high value of κ_{min} does not allow the mesh triangles to become too large even in the areas where the manifold curvature is relatively small. This results in a good quality approximation of the manifold which agrees very well with the 2D computations (green curves). The 2D computations were performed in an invariant plane $(x = 0, y, z)$, using an algorithm based on [51]. (The 2D contours are made up of approximately 10^4 points.) Figure 11 shows a comparison of computations performed for the same manifold with two different mesh resolutions. The red mesh corresponds to $R = 5$, as in figure 10, and the blue mesh corresponds to $R = 3$. Of course, a more thorough investigation of the algorithm accuracy and robustness is needed but we deem it beyond the scope of this publication and we postpone it to a subsequent study concerned with applications and performance of the algorithm.

[13] An instantaneous stagnation point (ISP) at time t^* satisfies $\boldsymbol{v}(\boldsymbol{x}_{isp}, t) = 0$; see also (98)

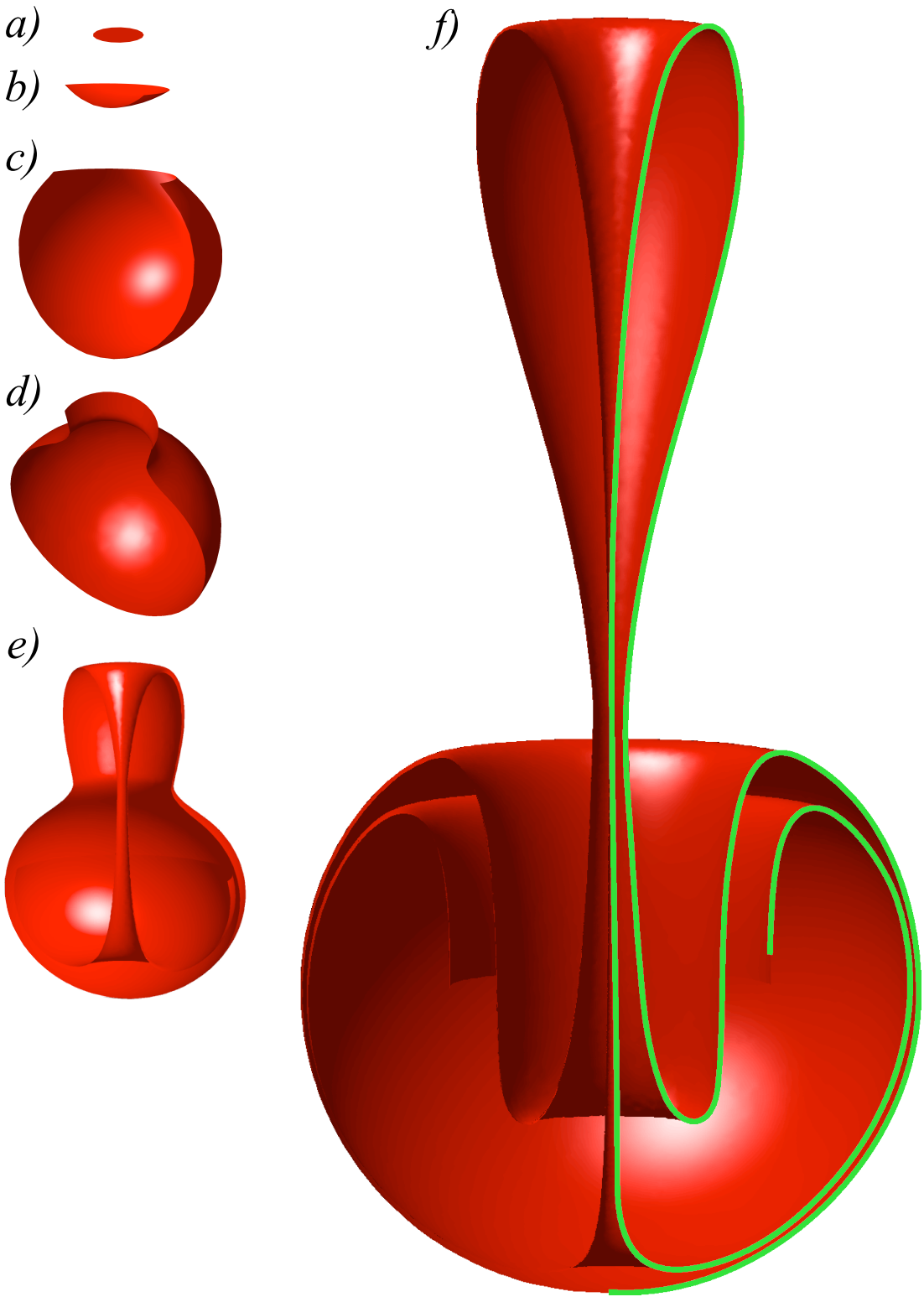

Figure 10: An ordered sequence of snapshots of the unstable manifold (red) of the the DHT $\boldsymbol{\gamma}_1$ present in the axisymmetric flow (79) with $\theta = 0$; cross-sections are shown at $(x = 0, y, z)$. The manifold snapshots were computed every $\Delta t = 0.1$, starting from a small disc ((a) not up to scale) centred around $\boldsymbol{\gamma}_1(t_0)$ at $t_0 = 0$; the snapshots shown correspond to a) $t = 0$, b) $t = 2.9$, c) $t = 4.7$, d) $t = 6.8$, e) $t = 7.1$, f) $t = 8.6$. The cross-section of the manifold snapshot shown in f) is represented using a mesh containing approximately 3.7×10^4 vertices and over 7.5×10^4 triangles. The green curve shows results obtained via 2D computations using an algorithm derived from [51].

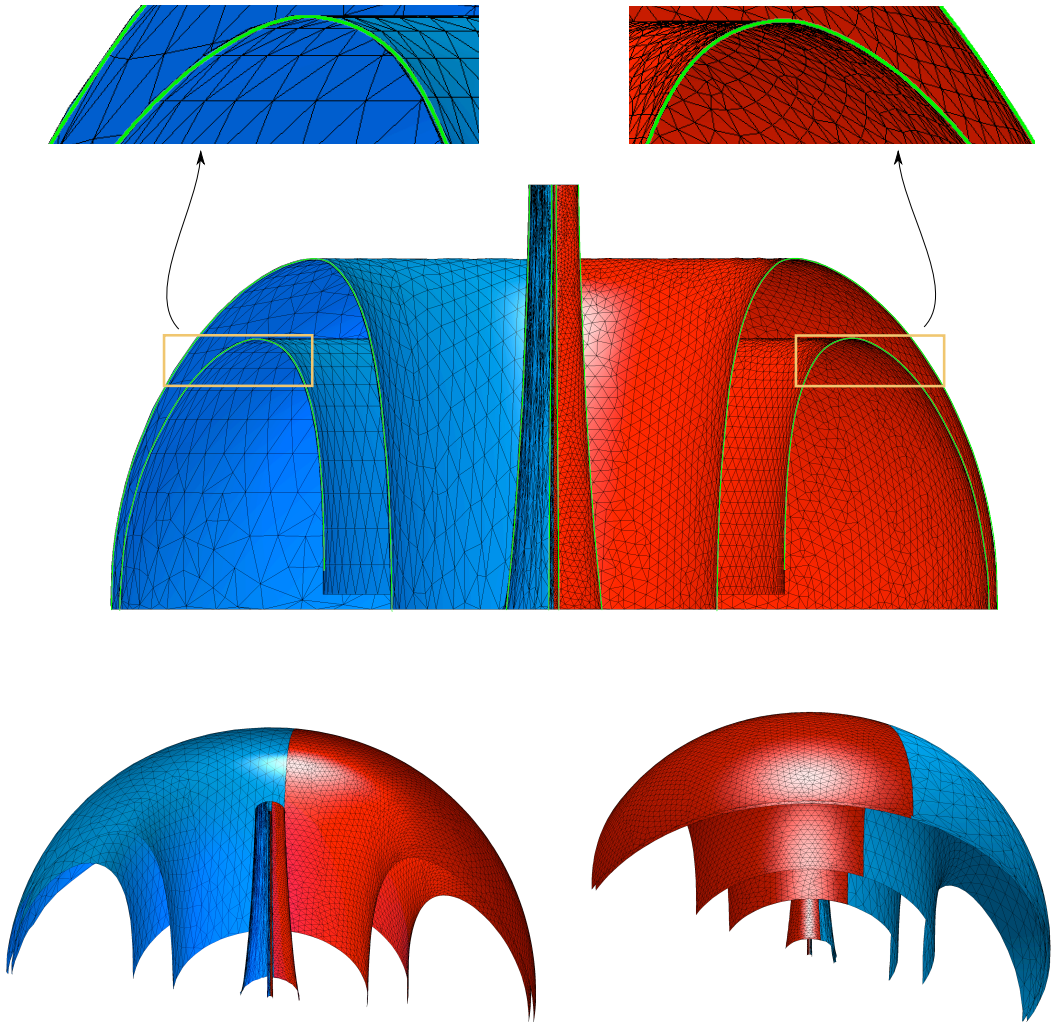

Figure 11: Comparison of surface meshes used to approximate the instantaneous manifold geometry at different resolutions (three different 3D views shown). The edge length is adapted according to the local curvature and the resolution parameter R (cf. (64)). Note, in particular, the difference in triangle size between the outer shells and the column in the centre. The red surface mesh ($R = 5, \kappa_{min} = 3, \kappa_{max} = 150$), which was used to represent the manifold snapshot in figure 10f, is shown after a completed refinement process and after two smoothing passes (cf. §5.2). The blue surface mesh ($R = 3, \kappa_{min} = 3, \kappa_{max} = 100$) is refined but not regularised. The green curve shows results obtained via 2D computations as in figure 10f.

Performing the computations with a smaller value of κ_{min} while keeping the same value of the resolution R, reduces the number of triangles in the low curvature areas. While this obviously decreases the computational time, it leads to larger interpolation errors when refining the mesh after advection. The balance between R, κ_{min} and κ_{max} is problem dependent but it is often possible to obtain satisfactory results even for smaller values of κ_{min}. Contrastingly, if the (problem-dependent) curvature cut-off value, κ_{max}, is set too small, consistent triangulation of the high-curvature regions becomes problematic. Computation of the presented results took about 48 hours ($R = 5$) and 36 hours ($R = 3$) of computer time, and were performed in MATLAB on a single PC running Windows XP with two Xeon 2.4 GHZ processors. We stress here that, since we were concerned at this stage with algorithm development, the code was unoptimised and not parallelised. Code optimisation is certainly important for practical applications and it will be dealt with subsequently. Consequently, a significant improvement in computational speed is expected.

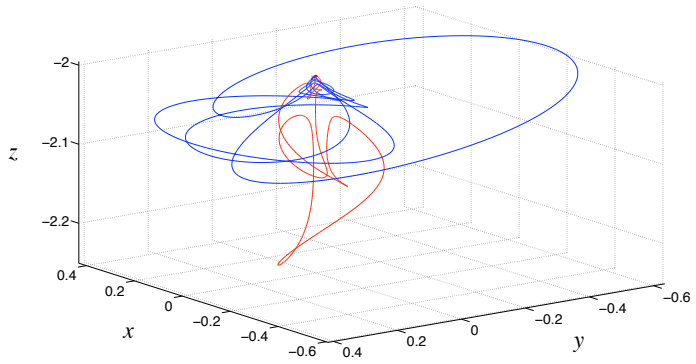

Figure 12: Geometry (projected onto $\mathbb{R}^3$) of the path of Instantaneous Stagnation Points (ISPs, blue) and the Distinguished Hyperbolic Trajectory (DHT, red) present in the 3D time-aperiodic flow discussed in (§6.1.2). There are two DHTs present in the flow (86), i.e. $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_2$, which are computed using the algorithm described in Appendix A. For clarify, only $\boldsymbol{\gamma}_1$ and the path of ISPs used to compute this DHT are shown here. The DHT $\boldsymbol{\gamma}_1$ has a three-dimensional unstable manifold (in the extended phase space); a snapshot of this manifold, given by a 2D surface, is shown in figure 13. Note that, since the flow was chosen to be asymptotically steady (see 86 and (87) shown in figure 9c)), both the ISP path and the DHT shown are closed curves. They start and end on the hyperbolic fixed point of the (steady) Hill's spherical vortex flow $h_1 = (0, 0, -2)$, (cf. 83). (The DHT $\boldsymbol{\gamma}_2$ starts and ends on the hyperbolic fixed point $h_1 = (0, 0, 2)$.) The asymptotic steadiness of the considered flow enables verification of the DHT computations, since for steady flows the projection of a DHT onto $\mathbb{R}^3$ must coincide with a hyperbolic fixed point in the flow.

6.1.2 Manifold computation in a three-dimensional unsteady flow

When $\theta \neq 0$ in (85), the velocity field (79) is fully three-dimensional. We study here one such an example where we choose the strain amplitude to be (see figure 9c)

$$\mathcal{A}(t) = \big(0.3 + 0.27\sin(3.3t)\big)\exp(-(t-6)^2/2.5^2), \tag{87}$$

and set the rotation parameters (in (85)) to

$$\theta = 0.5 + 0.05\sin 2t, \quad \varphi = 5t, \quad \psi = 0. \tag{88}$$

The particular choice of the strain amplitude implies that the resulting flow is asymptotically steady (and the underlying dynamical system is asymptotically autonomous), similarly to the axisymmetric case discussed earlier. There are two non-bifurcating hyperbolic ISPs in the flow throughout the evolution. Contrary to the axisymmetric flow configuration discussed in the previous section, the ISPs are no longer confined to the (flow-invariant) axis $\boldsymbol{e}_z$ but, as long as the perturbation is small, they remain in the neighbourhood of the hyperbolic stagnation points, h_1 and h_2, of the unperturbed Hill's spherical vortex. We note again that the 'smallness' of the perturbation amplitude is not a necessary condition for a successful computation of the DHTs and their manifolds but it provides a configuration which is easier to understand in these preliminary examples. As in the previous section, we use paths of the ISPs as the initial guesses in the DHT-finding algorithm, although other choices of the initial guess are possible (see Appendix A). The geometry (in $\mathbb{R}^3$) of one such DHT, namely $\boldsymbol{\gamma}_1$ (red), which was computed from ISP_1 (blue) is shown for this flow in figure 12. Note that, due to the particular choice of the strain amplitude (i.e. (87)), ISP_1 and $\boldsymbol{\gamma}_1$ start and end at the stagnation point h_1 (83) of the steady Hill's vortex flow. Similarly, ISP_2 and $\boldsymbol{\gamma}_2$ start and end at the stagnation point h_2, although this is not shown for clarity. The asymptotic steadiness of the considered flow enables verification of the DHT computations, since for steady flows the projection of a DHT onto $\mathbb{R}^3$ must coincide with a hyperbolic fixed point in the flow.

Similarly to the situation discussed in the previous section, $\boldsymbol{\gamma}_1$ has a two-dimensional unstable manifold but its snapshots, however, are no longer axisymmetric. Figure 13 shows an instantaneous geometry of the unstable manifold of $\boldsymbol{\gamma}_1$ at $t = 7.8$ which was 'grown', using our 3D algorithm, from a small disc containing $\boldsymbol{\gamma}_1(0)$ at $t = 0$; the parameters used were

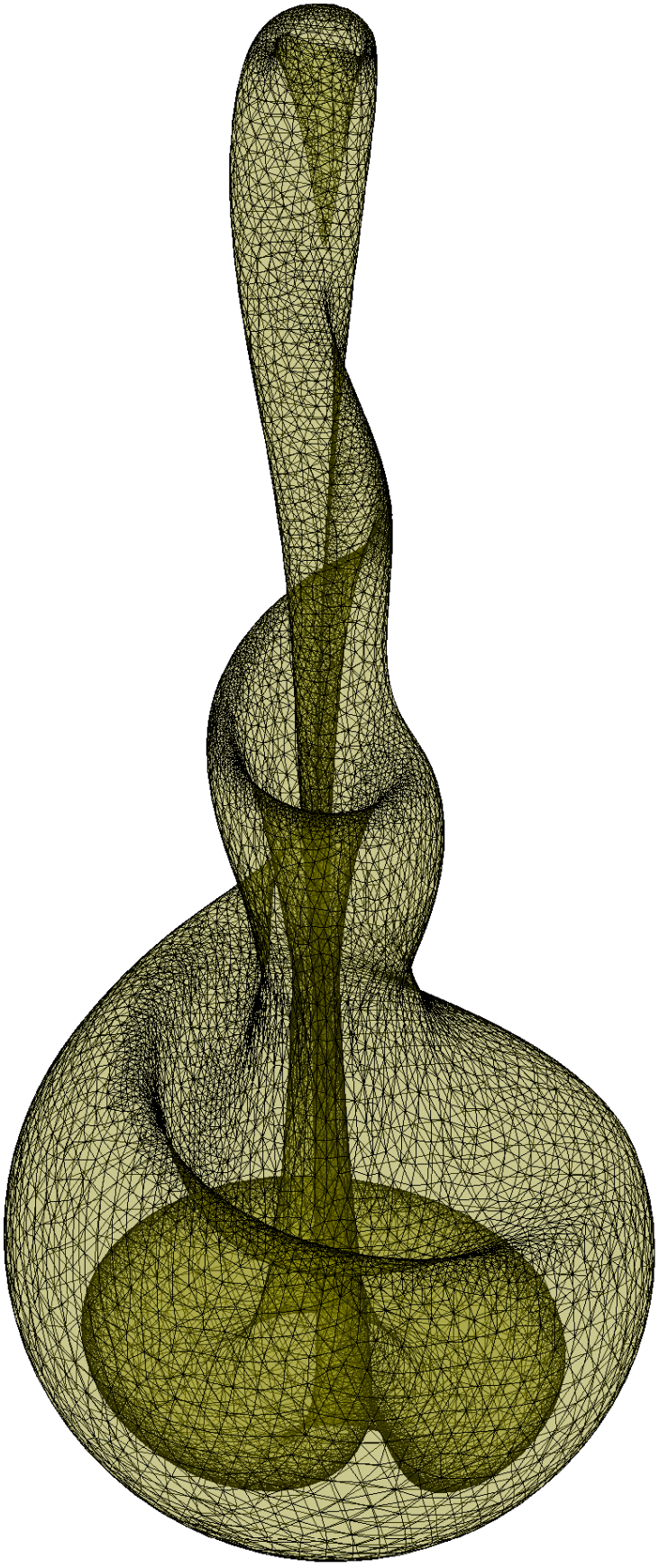

Figure 13: Instantaneous geometry (referred to as snapshot) of the unstable manifold of the the DHT $\boldsymbol{\gamma}_1$ (cf. figure 12) present in the flow (86) with a three-dimensional perturbation given in §6.1.2 at time $t = 7.8$. The surface shown is isotopic to a disc which was contained in the unstable subspace of the hyperbolic trajectory $\boldsymbol{\gamma_1}$ at $t = 0$ (cf. Appendix A); computations were performed with $\Delta t = 0.1$. The faces of the approximating mesh are semi-transparent in order to reveal the internal structure of the snapshot.

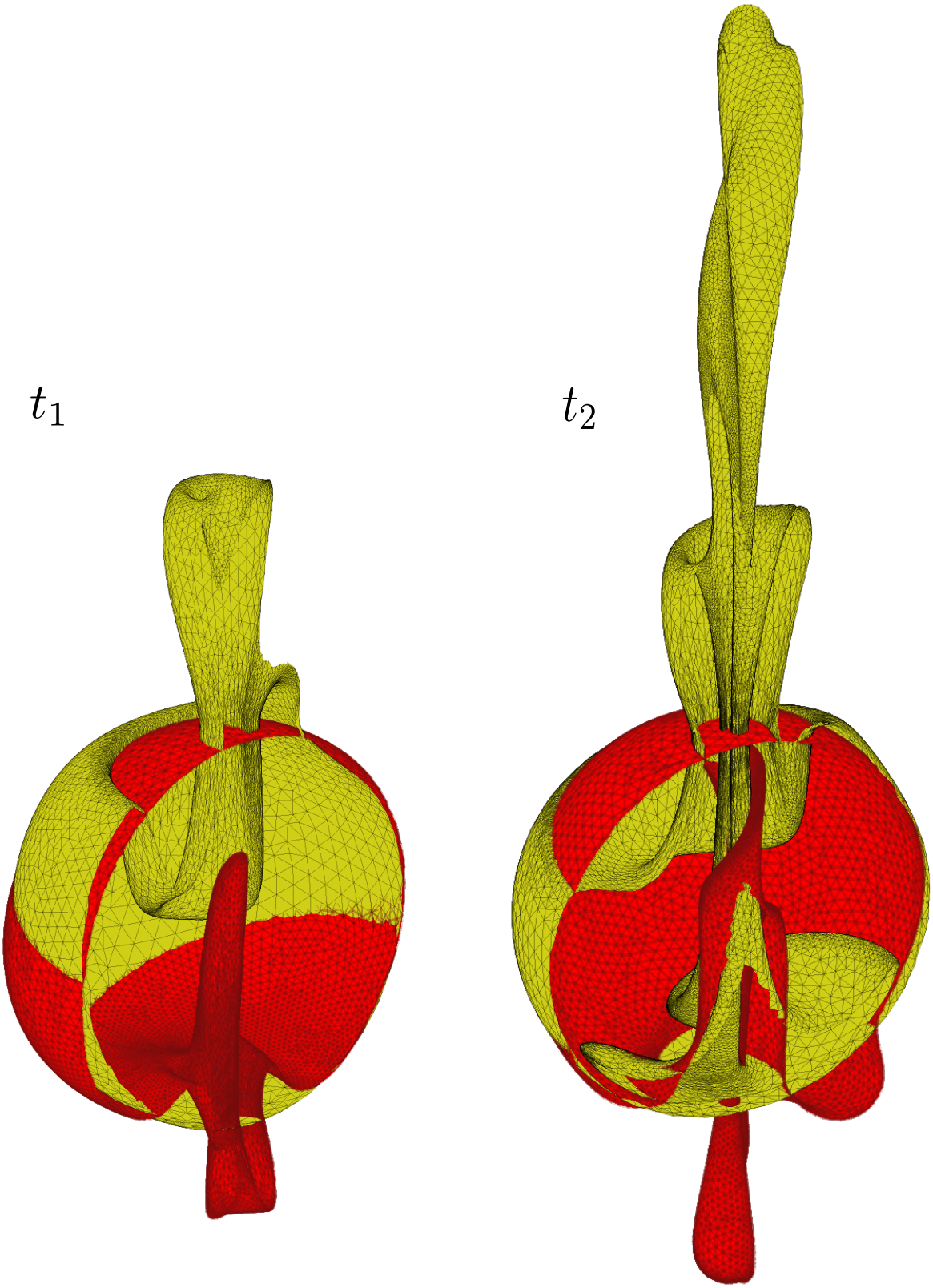

Figure 14: Cross-sections (at two different times $t_1 = 5.6$ and $t_2 = 7.8$) of the tangle of two-dimensional snapshots of the stable manifold (red) of the DHT $\boldsymbol{\gamma}_2$ and of the unstable manifold (olive green) of $\boldsymbol{\gamma}_1$ which are present in the three-dimensional unsteady flow (86) (see §6.1.2 for more details). Further development of numerical methods for accurate detection of manifold intersections, combined with analytical generalisation (and extension) of the tools developed in the context of *lobe dynamics* in the 2D setting, should enable the analysis of Lagrangian transport in 3D aperiodically time-dependent fluid flows.

$R = 4.5, \kappa_{min} = 2, \kappa_{max} = 150$. Note that the smaller value of κ_{min} used in these computations leads to generation of larger triangles, when compared to the axisymmetric manifold shown in figure 10) (see red mesh in figure 11), in the low curvature regions of the manifold. Such parameter values also lead to a larger variation of triangle size across the manifold. The mesh faces in figure 13 were made semi-transparent in order to allow visualisation of the internal structure of the manifold patch. It can be seen that the algorithm succeeds at adapting the mesh to the highly convoluted evolving geometry of the manifold snapshots by concentrating triangles in the areas of high curvature. The presented snapshot contains approximately 2.3×10^4 vertices and approximately 4.6×10^4 triangles. The full computation of the manifold snapshots between $t = 0$ and $t = 7.8$ took about 36 hours of computer time using the (unoptimised) MATLAB code on a single PC running Windows XP with two Xeon 2.4GHz processors. We expect the computational time to be significantly reduced once the code is optimised and parallelised.

Finally, in figure 14 we show cross-sections, at $(x, y = -0.3, z)$ and at two different time instants, of two snapshots[14] of the stable manifold of $\boldsymbol{\gamma}_2$ (red) and of the unstable manifold of $\boldsymbol{\gamma}_1$ (green). The intersections of the two manifolds form a complex invariant structure reminiscent of the invariant lobes in the 2D setting. Since the two triangulated manifolds represent approximations to invariant (material) surfaces in the flow, every volume bounded by the intersecting manifolds at some time t^* remains bounded by these manifold patches for all time. This suggests the possibility of extending the tools of lobe dynamics, developed in the two-dimensional context, to the three-dimensional case. However, the development of such tools is not straightforward and requires further careful considerations from both numerical and analytical perspective. In the concluding section below, we outline the most important developments which need to be implemented before achieving this goal.

7 Conclusions

In this paper we presented a numerical method for approximating the geometry of three-dimensional invariant manifolds in non-autonomous, three-dimensional dynamical systems. The developed algorithm, representing an invariant manifold as a time-ordered sequence of two-dimensional snapshots of its instantaneous geometry, combines an automatic mesh refinement with adaptive vertex redistribution and remeshing. This allows for a computationally efficient determination of highly convoluted, evolving geometry of the two-dimensional manifold snapshots in a fully 3D time-aperiodic setting. We stress again here that when an invariant manifold is time-independent (e.g. the stable and unstable manifolds in steady 3D flows; cf. §2.2) or when the underlying dynamical system depends periodically on time, algorithms which explicitly utilise these simplifications represent a more computationally efficient choice (cf. §1 and [19, 16, 8]).

In regard to fluid dynamical applications, we showed that the developed method is capable of providing detailed information on the evolving Lagrangian flow structure in three dimensions over long periods of time, and that it can be used to study the geometry of any compact, orientable and simply connected subset of an invariant manifold in 3D fluid flows with arbitrary time dependence. Thus, the algorithm can be used with equal success to track orientable material surfaces advected by the fluid flow, or to study the geometry of (finite-time) stable and unstable manifolds of (finite-time) hyperbolic trajectories. We also extended the method for computing the finite-time hyperbolic trajectories in 3D non-autonomous dynamical systems (with arbitrary time dependence) which are needed for initiating the manifold computations (cf. Appendix A).

The results presented here are only the first step towards the Lagrangian transport analysis in 3D unsteady fluid flows. As already pointed out at the beginning of this paper, a number of further developments have to be implemented in order to achieve such a goal. The most important steps of this process can be roughly divided into two classes and we briefly summarise them below:

Further analytical developments:

[14] As discussed earlier, the presented snapshots correspond to compact subsets (isotopic to a disc) of the snapshots of the respective invariant manifolds.

- *Establishment of an appropriate definition of a three-dimensional lobe.* In three dimensions, an intersection of invariant surfaces (or a self-intersection of a single surface) do not necessarily have to form a closed curve (i.e. a loop). If a pair of two-dimensional differentiable manifolds intersects along a loop, the volume bounded by the respective patches of the manifolds corresponds to a three-dimensional lobe. Careful analysis is needed in order to understand and quantify transport characteristics within the volumes which are formed by manifold intersections along an infinite-length curve (e.g. an infinite-volume 'spiraling' tube).

Forthcoming numerical developments:

- *Parallelisation of the code.* The developed code should be parallelised in order to maximise its efficiency and data handling. In particular, the advection of the mesh nodes and updating the mesh connectivity data can be significantly speeded up when performed in parallel, especially in the case of computations at high resolutions when the meshes consist of a large number of vertices (and triangles). Other algorithms, especially those involved in interpolating new manifold points, remeshing and regularising the mesh, are amenable to distribution over a computer grid or cluster by subdividing the meshes into smaller patches.
- *Detection of manifold intersections based on their triangular-mesh approximations.* Development of a method for detecting manifold intersections is needed for visualising the lobes (see above) and understanding details of transport in 3D flows.
- *Application of the software to flow data obtained from realistic numerical 3D models.* The Lagrangian transport associated with 3D mesoscale oceanic eddies is of particular interest.

The progress on the issues listed above will be communicated in future publications. A 'user friendly' version of the software, combining algorithms for 2D computations, as well as the code for 3D computations, is under development (the algorithms are being integrated with a GUI). The software will be made available online at `http://lacms.maths.bris.ac.uk/software/`.

Acknowledgements

The authors acknowledge financial support from ONR Grant No. N00014-01-1-0769, and the stimulating environment of the NSF sponsored Institute for Mathematics and its Applications (IMA) at the University of Minnesota, where this manuscript was completed.

A Computation of Distinguished Hyperbolic Trajectories in non-autonomous, three-dimensional dynamical systems

A numerical method for computing Distinguished Hyperbolic Trajectories in two-dimensional, non-autonomous dynamical systems was developed in [32, 35] and it was studied in detail in [49, 50]. Here, we extend this method to the three-dimensional, non-autonomous setting. Most of the analytical derivations needed for such an extension are identical with those developed in the 2D, non-autonomous context. However, since some significant changes appear at a later stage of the derivation, we recapitulate the most important notions below.

Let $I = [t_i, t_f] \subset \mathbb{R}$ be a time interval and let $\tilde{\boldsymbol{x}}(t) : I \to \mathbb{R}^3$ be a C^r ($r \geqslant 1$) curve in $\mathbb{R}^3$. Consider now a linearisation of the system (2) about $\tilde{\boldsymbol{x}}(t)$ given by

$$\dot{\boldsymbol{\xi}} = \partial_{\boldsymbol{x}} \boldsymbol{v}(\tilde{\boldsymbol{x}}(t), t)\, \boldsymbol{\xi}, \tag{89}$$

where $\boldsymbol{\xi} = \boldsymbol{x} - \tilde{\boldsymbol{x}}$ and $\partial_{\boldsymbol{x}} \boldsymbol{v}(\tilde{\boldsymbol{x}}(t), t)$ is the Jacobian of $\boldsymbol{v}(\boldsymbol{x}, t)$ evaluated at $\boldsymbol{x} = \tilde{\boldsymbol{x}}(t)$. We let $\mathbf{X}(t, t_i)$ denote the fundamental solution matrix of (89).

Definition A.1 (Finite-time Exponential Dichotomy)**.** We say that the linear equation (89) has an *exponential dichotomy* on the finite time interval I if there exists a projection operator

$\mathbf{P} : I\!R \to I\!R^3$, $\mathbf{P^2} = \mathbf{P}$, and positive constants K, L, α, β such that:

$$|\mathbf{X}(t,t_i)\mathbf{P}\mathbf{X}^{-1}(s,t_i)| \leqslant Ke^{-\alpha(t-s)}, \quad \text{for} \quad t \geqslant s, \quad t,s \in I,$$

$$|\mathbf{X}(t,t_i)(\mathbf{Id}-\mathbf{P})\mathbf{X}^{-1}(s,t_i)| \leqslant Le^{-\beta(s-t)}, \quad \text{for} \quad s \geqslant t, \quad t,s \in I. \tag{90}$$

In the limit $t_i \to -\infty$, $t_f \to \infty$, the above definition becomes equivalent to the standard notion of exponential dichotomy (see [12, 55, 30] for more details). Numerical methods for calculating the constants K, L, α, and β are given in [14].

Using the notion of exponential dichotomy, we can now specify what is meant by finite-time hyperbolicity.

Definition A.2 (Finite-time Hyperbolicity)**.** We say that the path $\tilde{\boldsymbol{x}}(t) : I \to I\!R^3$ is *finite-time hyperbolic* on the interval I if the equation (89) has exponential dichotomy on I. Furthermore, if $\boldsymbol{\gamma}(t)$ is a trajectory of the system (2) and we let $\tilde{\boldsymbol{x}} = \boldsymbol{\gamma}(t)$, then $\boldsymbol{\gamma}$ is called a *finite-time hyperbolic trajectory* on the interval I if the equation

$$\dot{\boldsymbol{\xi}} = \partial_{\mathbf{x}}\boldsymbol{v}(\boldsymbol{\gamma}(t),t)\,\boldsymbol{\xi}, \tag{91}$$

has exponential dichotomy on I.

Remark: In the limit $t_i \to -\infty$, $t_f \to \infty$, the above definition becomes equivalent to the standard notion of a hyperbolic trajectory.

Roughly speaking, finite-time hyperbolicity implies that trajectories located sufficiently close to $\boldsymbol{\gamma}(t)$ separate at an exponential rate either in forward or in backward time; no assumptions are made the about the fate of these neighbouring trajectories beyond I even if the velocity field $\boldsymbol{v}(\boldsymbol{x},t)$ is known outside this interval.

Definition A.3 (Frozen-time Hyperbolicity)**.** We say that the path $\tilde{\boldsymbol{x}}(t)$ is frozen-time hyperbolic on the finite interval I if the eigenvalues of the Jacobian, $\partial_{\mathbf{x}}\boldsymbol{v}(\tilde{\boldsymbol{x}}(t),t)$, in the linearised equation (89) have non-zero real parts for any fixed $t \in I$.

Remark: It can be shown, using results discussed in [36], that a path which is frozen-time hyperbolic is also finite-time hyperbolic (but not vice versa).

Using definitions A.1 and A.2, we can finally identify a special type of a hyperbolic trajectory which, if present in a considered flow, plays an important role in Lagrangian transport considerations.

Let $\tilde{\boldsymbol{x}}(t)$ be a finite-time hyperbolic path and consider the nonlinear equation (2) in a frame 'moving' with $\tilde{\boldsymbol{x}}$ by setting $\boldsymbol{x}(t) = \boldsymbol{y}(t) + \tilde{\boldsymbol{x}}(t)$. The transformed equation can be written as

$$\dot{\boldsymbol{y}} = \mathbf{A}(t)\boldsymbol{y} + \mathbf{f}(\boldsymbol{y},t), \qquad \boldsymbol{y} \in I\!R^3,\ t \in I. \tag{92}$$

where

$$\mathbf{A}(t) = \partial_{\mathbf{x}}\boldsymbol{v}(\tilde{\boldsymbol{x}}(t),t), \tag{93}$$

$$\mathbf{f}(\boldsymbol{y},t) = \boldsymbol{v}\big(\boldsymbol{y}(t)+\tilde{\boldsymbol{x}}(t)\big) - \partial_{\mathbf{x}}\boldsymbol{v}\big(\tilde{\boldsymbol{x}}(t),t\big)\boldsymbol{y}(t) - \dot{\tilde{\boldsymbol{x}}}(t). \tag{94}$$

Since we assumed that $\tilde{\boldsymbol{x}}(t)$ is finite-time hyperbolic (see Definition A.2), we can associate the particular solution of (92) with the following integral equation

$$\boldsymbol{y}(t) = \mathbf{X}(t,t_i)\int_{t_i}^{t}\mathbf{P}\mathbf{X}^{-1}(s,t_i)\mathbf{f}(\boldsymbol{y}(s),s)\mathrm{d}s - \mathbf{X}(t,t_i)\int_{t}^{t_f}(\mathbf{Id}-\mathbf{P})\mathbf{X}^{-1}(s,t_i)\mathbf{f}(\boldsymbol{y}(s),s)\mathrm{d}s, \tag{95}$$

where $\mathbf{P}$ is the projection operator associated with the exponential dichotomy (90) and $\mathbf{X}$ is the fundamental solution matrix associated with the linear part of (92). It can be easily checked that the solution of (95) represents the only solution of (92) which does not exhibit exponential growth or decay within I. Furthermore, using very similar techniques to those employed in [36], it can be shown that, for given $\tilde{\boldsymbol{x}}(t)$, the solution of (95) is finite-time hyperbolic and unique on the time interval I provided that

$$||\boldsymbol{v}(\boldsymbol{x}(t),t) - \partial_{\mathbf{x}}\boldsymbol{v}(\tilde{\boldsymbol{x}}(t),t)(\boldsymbol{x}(t)-\tilde{\boldsymbol{x}}(t)) - \dot{\tilde{\boldsymbol{x}}}(t)||_\infty < \infty, \qquad \forall \quad t \in I, \tag{96}$$

and

$$||\partial_{\mathbf{x}}\boldsymbol{v}(\boldsymbol{x}(t),t) - \partial_{\mathbf{x}}\boldsymbol{v}(\tilde{\boldsymbol{x}}(t),t)||_\infty < \left(\frac{K}{\alpha}+\frac{L}{\beta}\right)^{-1}, \qquad \forall \quad t \in I. \tag{97}$$

The constants K, L, α, β are associated with the exponential dichotomy of the linear part of (92) (cf. Definition A.1).

Definition A.4 (Distinguished, Finite-time Hyperbolic Trajectory)**.** Let $\tilde{\boldsymbol{x}}(t)$ be a finite-time hyperbolic path which does not have an exponential component within I. A trajectory $\boldsymbol{\gamma}(t)$ of the system (2) is called a *Finite-time Distinguished Hyperbolic Trajectory* if it can be written as $\boldsymbol{\gamma}(t) = \boldsymbol{y}(t) + \tilde{\boldsymbol{x}}(t)$ where $\boldsymbol{y}(t)$ satisfies the integral equation (95) subject to the conditions (96) and (97).

Note that the finite-time hyperbolic path $\tilde{\boldsymbol{x}}(t)$ used in Definition A.4 can be given, in particular, by a path of *Instantaneous stagnation points* (ISPs) which are frozen-time hyperbolic (cf. Definition A.3). Given the velocity field (1), a path of ISPs is given by a continuous curve, $\boldsymbol{x}_{isp}(t)$, such that

$$\boldsymbol{v}(\boldsymbol{x}_{\rm isp}(t),t) = 0, \qquad t \in \tilde{I}, \tag{98}$$

where $\tilde{\mathcal{T}} \subset I$ is a time interval within which the Jacobian, $\partial_{\mathbf{x}}\boldsymbol{v}(\boldsymbol{x}_{isp}(t),t)$, does not vanish, as required by the Implicit Function Theorem for the existence of a solution to (98).

A.1 Numerical implementation of the algorithm

Even if the path $\tilde{\boldsymbol{x}}(t)$ is known, determination of a DHT by solving (95) is prohibitive. Instead, following [32], it can be shown that there exists a time-dependent linear transformation $\mathscr{T} : I\!R^3 \to I\!R^3$ such that

$$\boldsymbol{w} = \mathscr{T}(t)\boldsymbol{y}, \tag{99}$$

where

$$\mathscr{T}(t) = e^{(t-t_i)\mathbf{D}}\mathbf{R}^T(t_f,t_i)\mathbf{R}(t,t_i)e^{-\mathbf{\Sigma}(t,t_i)}\mathbf{B}^T(t,t_i). \tag{100}$$

The matrices $\mathbf{B}(t,t_i)$, $\mathbf{R}(t,t_i)$, and $\mathbf{\Sigma}(t,t_i)$ in (100) correspond to the SVD decomposition of the fundamental solution matrix, $\mathbf{X}(t,t_i)$, i.e.

$$\mathbf{X}(t,t_i) = \mathbf{B}(t,t_i)e^{\mathbf{\Sigma}(t,t_i)}\mathbf{R}(t,t_i)^T, \tag{101}$$

and the constant diagonal matrix $\mathbf{D}$ is given by

$$\mathbf{D} = \mathbf{\Sigma}(t_f,t_i)/(t_f-t_i). \tag{102}$$

A system describing the evolution of the matrices $\mathbf{B}$, $\mathbf{\Sigma}$ and $\mathbf{R}$, is derived in §A.2. Applying the transformation (99) to (92) leads to

$$\dot{\boldsymbol{w}} = \mathbf{D}\,\boldsymbol{w} + \mathbf{g}(\boldsymbol{w},t). \tag{103}$$

where $\mathbf{g}(\boldsymbol{w},t)$ is given by

$$\mathbf{g}(\boldsymbol{w},t) = \mathscr{T}(t)\,\mathbf{f}\big(\mathscr{T}^{-1}(t)\,\boldsymbol{w},t\big) \tag{104}$$

$$= \mathscr{T}(t)\,\boldsymbol{v}\big(\mathscr{T}^{-1}(t)\,\boldsymbol{w}+\tilde{\boldsymbol{x}}(t)\big) - \mathscr{T}(t)\partial_{\mathbf{x}}\boldsymbol{v}\big(\tilde{\boldsymbol{x}}(t),t\big)\mathscr{T}^{-1}(t)\,\boldsymbol{w} - \mathscr{T}(t)\,\dot{\tilde{\boldsymbol{x}}}(t). \tag{105}$$

Using (99) in (95) transforms the integral equation into a much simplified form

$$\boldsymbol{w}(t) = \begin{bmatrix} \chi(-d_1)\int_{t_i}^{t} e^{d_1(t-s)}g_1(\boldsymbol{w}(s),s)\mathrm{d}s - \chi(d_1)\int_t^{t_f} e^{d_1(t-s)}g_1(\boldsymbol{w}(s),s)\mathrm{d}s \\ \chi(-d_2)\int_{t_i}^{t} e^{d_2(t-s)}g_1(\boldsymbol{w}(s),s)\mathrm{d}s - \chi(d_2)\int_t^{t_f} e^{d_2(t-s)}g_2(\boldsymbol{w}(s),s)\mathrm{d}s \\ \chi(-d_3)\int_{t_i}^{t} e^{d_3(t-s)}g_1(\boldsymbol{w}(s),s)\mathrm{d}s - \chi(d_3)\int_t^{t_f} e^{d_3(t-s)}g_3(\boldsymbol{w}(s),s)\mathrm{d}s \end{bmatrix}, \tag{106}$$

where (d_1,d_2,d_3) are the eigenvalues of $\mathbf{D}$ and the step function χ satisfies

$$\chi(x) = \begin{cases} 1 \ \text{ if } \ x > 0, \\ 0 \ \text{ otherwise.} \end{cases} \tag{107}$$

Given a finite-time hyperbolic path $\tilde{\boldsymbol{x}}(t)$, a solution to (106) can be found numerically in an

iterative fashion which is based on the following analytical arguments:

Consider a (Banach) space of all bounded and continuous functions defined on $\mathbb{R}$ with values in $\mathbb{R}^3$, denoted by $BC[\mathbb{R}, \mathbb{R}^3]$, and define the following map from $BC[\mathbb{R}, \mathbb{R}^3]$ to itself

$$\mathcal{T}\boldsymbol{w} = \begin{bmatrix} \chi(-d_1)\int_{t_i}^{t} e^{d_1(t-s)} g_1(\boldsymbol{w}(s), s)\mathrm{d}s - \chi(d_1)\int_{t}^{t_f} e^{d_1(t-s)} g_1(\boldsymbol{w}(s), s)\mathrm{d}s \\ \chi(-d_2)\int_{t_i}^{t} e^{d_2(t-s)} g_1(\boldsymbol{w}(s), s)\mathrm{d}s - \chi(d_2)\int_{t}^{t_f} e^{d_2(t-s)} g_2(\boldsymbol{w}(s), s)\mathrm{d}s \\ \chi(-d_3)\int_{t_i}^{t} e^{d_3(t-s)} g_1(\boldsymbol{w}(s), s)\mathrm{d}s - \chi(d_3)\int_{t}^{t_f} e^{d_3(t-s)} g_3(\boldsymbol{w}(s), s)\mathrm{d}s \end{bmatrix}. \tag{108}$$

Clearly, the solution of (106) is given by the fixed point of (108), i.e. $\mathcal{T}\boldsymbol{w} = \boldsymbol{w}$. Using the same techniques as in [36], it can be shown that, similarly to (95), the $\mathcal{T}$ is a contraction map. Consequently, based on the contraction mapping principle, for any initial $\boldsymbol{w}_0(t)$ such that $\boldsymbol{x}(t) = \mathscr{T}^{-1}(t)\boldsymbol{w}_0(t) + \tilde{\boldsymbol{x}}(t)$ satisfies (96) and (97), the solution of (106) is finite-time hyperbolic and is given by the limit

$$\boldsymbol{w} = \lim_{n\to\infty} \mathcal{T}^n \boldsymbol{w}_0. \tag{109}$$

The numerical procedure can be briefly described as follows:

1) Given the finite-time hyperbolic path $\tilde{\boldsymbol{x}}$, set $\boldsymbol{w}^0 = 0$.
2) Calculate $\boldsymbol{w}^1$ by evaluating (108) on $\boldsymbol{w}^0$, i.e. $\boldsymbol{w}^1 = \mathcal{T}\boldsymbol{w}_0$.
3) Repeat the procedure, using $\boldsymbol{w}^{(n+1)} = \mathcal{T}\boldsymbol{w}^{(n)} = \mathcal{T}^n\boldsymbol{w}^{(0)}$, until $\boldsymbol{w}^{n+1} \approx \boldsymbol{w}^n$.
4) Transform the solution back to the $\boldsymbol{x}$ coordinates by evaluating $\boldsymbol{x}(t) = \mathscr{T}^{-1}(t)\boldsymbol{w}(t) + \tilde{\boldsymbol{x}}(t)$.

Note finally that contrary to the method used in [49, 50], the transformation (99) is determined only once and the whole procedure relies on the exponential dichotomy associated with linearization of (92) about $\tilde{\boldsymbol{x}}(t)$. This approach justifies the use of the iterative procedure due to the contractive properties of a single map (108), which is in line with the results discussed in [36].

A.2 Evolution equations for $\mathbf{B}(t)$, $\mathbf{\Sigma}(t)$ and $\mathbf{R}(t)$.

In order to compute the transformation matrix (101) obtained in §A.1 we need to determine the evolution of the orthogonal matrices $\mathbf{B}$, $\mathbf{R}$ and of the diagonal matrix $\mathbf{\Sigma}$. Since these matrices correspond to the SVD decomposition of the fundamental solution matrix $\mathbf{X}$, a straightforward way of computing $\mathbf{B}(t)$, $\mathbf{R}(t)$, $\mathbf{\Sigma}(t)$ is to first compute the fundamental solution matrix, which satisfies

$$\dot{\mathbf{X}} = \mathbf{F}(t)\,\mathbf{X}, \tag{110}$$

and then perform the SVD decomposition at each required time instant t. We note, however, that solutions of (110) may experience exponential growth and attempts to solve this system directly may result in overflowing the machine precision. Therefore, $\mathbf{X}$ is determined explicitly in terms of the matrices $\mathbf{B}$, $\mathbf{\Sigma}$, $\mathbf{R}$ by substituting (101) into (110) which gives

$$\dot{\mathbf{B}}\exp(\mathbf{\Sigma})\mathbf{R}^T + \mathbf{B}\dot{\mathbf{\Sigma}}\exp(\mathbf{\Sigma})\mathbf{R}^T + \mathbf{B}\exp(\mathbf{\Sigma})\dot{\mathbf{R}}^T = \mathbf{F}\mathbf{B}\exp\mathbf{\Sigma}, \tag{111}$$

which, after multiplying (111) from the left by $\mathbf{B}^T$ and from the right by $\mathbf{R}\exp(-\mathbf{\Sigma})$, can be cast in a more suitable form

$$\mathbf{B}^T\dot{\mathbf{B}} + \dot{\mathbf{\Sigma}} + \exp(\mathbf{\Sigma})\dot{\mathbf{R}}^T\mathbf{R}\exp(-\mathbf{\Sigma}) = \mathbf{H}, \tag{112}$$

where $\mathbf{H} = \mathbf{B}^T\mathbf{F}\mathbf{B}$.

Since the matrices $\mathbf{B}$ and $\mathbf{R}$ are orthogonal, they can be parameterised as

$$\mathbf{B} = \begin{bmatrix} \cos\psi\cos\varphi - \sin\psi\cos\theta\sin\varphi & \cos\psi\sin\varphi + \sin\psi\cos\theta\cos\varphi & \sin\psi\sin\theta \\ -\sin\psi\cos\varphi - \cos\psi\cos\theta\sin\varphi & -\sin\psi\sin\varphi + \cos\psi\cos\theta\cos\varphi & \cos\psi\sin\theta \\ \sin\theta\sin\varphi & -\sin\theta\cos\varphi & \cos\theta \end{bmatrix}, \tag{113}$$

and

$$\mathbf{R} = \begin{bmatrix} \cos\gamma\cos\alpha - \sin\gamma\cos\beta\sin\alpha & \cos\gamma\sin\alpha + \sin\gamma\cos\beta\cos\alpha & \sin\gamma\sin\beta \\ -\sin\gamma\cos\alpha - \cos\gamma\cos\beta\sin\alpha & -\sin\gamma\sin\alpha + \cos\gamma\cos\beta\cos\alpha & \cos\gamma\sin\beta \\ \sin\beta\sin\alpha & -\sin\beta\cos\alpha & \cos\beta \end{bmatrix}, \tag{114}$$

where the triples (ψ, φ, θ) and (γ, α, β) represent two sets of Euler angles defined in the standard way. Substitution of (113) and (114) into (112) leads to the following system of nine equations for the parameters $(\sigma_1, \sigma_2, \sigma_3, \psi, \phi, \theta, \gamma, \alpha, \beta)$:

$$\dot{\sigma}_1 = H_{11} \tag{115}$$

$$\dot{\sigma}_2 = H_{22} \tag{116}$$

$$\dot{\sigma}_3 = H_{33} \tag{117}$$

$$\begin{aligned}\dot{\psi} = \frac{1}{4}\Big[&(H_{32}\sin\phi + H_{31}\cos\phi)e^{\sigma_2-2\sigma_3+\sigma_1} - (H_{13}\cos\phi + H_{23}\sin\phi)e^{2\sigma_3-\sigma_2-\sigma_1} \\ &+ (H_{23}\sin\phi - H_{31}\cos\phi)e^{\sigma_1-\sigma_2} \\ &+ (H_{13}\cos\phi - H_{32}\sin\phi)e^{\sigma_2-\sigma_1}\Big] \Big/ \Big[\sin\theta\sinh(\sigma_1-\sigma_3)\sinh(\sigma_3-\sigma_2)\Big]\end{aligned} \tag{118}$$

$$\begin{aligned}\dot{\phi} = \frac{1}{8}\Big[&((H_{31}\cos\phi - H_{23}\sin\phi)\cos\theta - H_{21}\sin\theta)e^{2(\sigma_1-\sigma_2)} \\ &- ((H_{32}\sin\phi + H_{31}\cos\phi)\cos\theta - H_{21}\sin\theta)e^{2(\sigma_1-\sigma_3)} \\ &+ ((H_{23}\sin\phi + H_{13}\cos\phi)\cos\theta + H_{21}\sin\theta)e^{2(\sigma_3-\sigma_2)} \\ &- ((H_{23}\sin\phi + H_{13}\cos\phi)\cos\theta - H_{12}\sin\theta)e^{2(\sigma_3-\sigma_1)} \\ &+ ((H_{32}\sin\phi + H_{31}\cos\phi)\cos\theta + H_{12}\sin\theta)e^{2(\sigma_2-\sigma_3)} \\ &+ ((H_{13}\cos\phi - H_{32}\sin\phi)\cos\theta - H_{12}\sin\theta)e^{2(\sigma_2-\sigma_1)} \\ &+ ((H_{23}+H_{32})\sin\phi - (H_{31}+H_{13})\cos\phi)\cos\theta - (H_{21}+H_{12})\sin\theta\Big] \\ &\Big/\Big[\sin\theta\sinh(-\sigma_1+\sigma_2)\sinh(\sigma_3-\sigma_1)\sinh(\sigma_3-\sigma_2))\Big]\end{aligned} \tag{119}$$

$$\begin{aligned}\dot{\theta} = \frac{1}{4}\Big[&(H_{23}\cos\phi - H_{13}\sin\phi)e^{2\sigma_3-\sigma_2-\sigma_1} + (H_{31}\sin\phi - H_{32}\cos\phi)e^{\sigma_2-2\sigma_3+\sigma_1} \\ &- (H_{23}\cos\phi + H_{31}\sin\phi)e^{\sigma_1-\sigma_2} \\ &+ (H_{13}\sin\phi + H_{32}\cos\phi)e^{\sigma_2-\sigma_1}\Big] \Big/ \Big[\sinh(\sigma_3-\sigma_1)\sinh(\sigma_3-\sigma_2)\Big]\end{aligned} \tag{120}$$

$$\begin{aligned}\dot{\alpha} = 4\Big[&(H_{23}+H_{32})\cos\beta\sin\alpha\,(\cosh(2\sigma_1-\sigma_3-\sigma_2) - \cosh(\sigma_2-\sigma_3)) \\ &+ (H_{21}+H_{12})\sin\beta\,(\cosh(\sigma_1-\sigma_2) - \cosh(2\sigma_3-\sigma_2-\sigma_1)) \\ &+ (H_{31}+H_{13})\cos\beta\cos\alpha\,(\cosh(\sigma_1-\sigma_3) - \cosh(2\sigma_2-\sigma_3-\sigma_1))\Big] \\ &\Big/\Big[\sin\beta\sinh(\sigma_1-\sigma_3)\sinh(\sigma_3-\sigma_2))\sinh(\sigma_2-\sigma_1))\Big]\end{aligned} \tag{121}$$

$$\dot{\beta} = \frac{1}{2}\Big((H_{23}+H_{32})\cos\alpha\,\mathrm{cosech}(\sigma_3-\sigma_2) + (H_{31}+H_{13})\sin\alpha\,\mathrm{cosech}(\sigma_1-\sigma_3)\Big) \tag{122}$$

$$\dot{\gamma} = \frac{\mathrm{cosec}\beta}{2}\Big((H_{13}+H_{31})\cos\alpha\,\mathrm{cosech}(\sigma_3-\sigma_1) + (H_{23}+H_{32})\sin\alpha\,\mathrm{cosech}(\sigma_3-\sigma_2)\Big). \tag{123}$$

B Adaptive advection of manifold boundary

As discussed §5.3, if remeshing of a manifold snapshot $\mathcal{M}_{t_{n+1}} \subset \Omega \subset I\!R^3$ is necessary, an initial front $\mathfrak{S} \subset \mathcal{M}_{t_{n+1}}$ has to be constructed from which the triangulation is extended over the whole $\mathcal{M}_{t_{n+1}}$. If the computed two-dimensional manifold snapshots $\mathcal{M}_{t_{n+1}}$ have a boundary, $\partial\mathcal{M}_{t_{n+1}}$, the initial front is constructed (see §5.3) from the high-resolution discrete copy of the boundary, $\mathfrak{B}_{t_{n+1}}$, which is evolved independently of the mesh $(\mathfrak{M}_{t_{n+1}}, \mathbb{M}^{t_{n+1}})$ approximating $\mathcal{M}_{t_{n+1}}$ (see §3.1). If the manifold snapshots are represented by closed surfaces, a closed curve $\mathfrak{L}_{t_{n+1}} \subset \mathcal{M}_{t_{n+1}}$ (cf. §3.1), is used to construct the initial front. In both these cases, a robust and accurate method is needed for determining the geometry of the sequences $\{\mathfrak{B}_{t_n}\}_{n\in Z_N}$ or $\{\mathfrak{L}_{t_n}\}_{n\in Z_N}$ during the evolution governed by the family of diffeomorphisms $\{\phi_{t_n}\}_{t_n\in I}, n \in Z_N$ (cf. (9)). We provide here more details on the adaptive numerical method used for computing the evolving geometry of a closed piecewise C^2 curve in $I\!R^3$ under the dynamics induced $\{\phi_\tau\}_{\tau\in I}$.

The numerical procedure is based on a 3D generalisation of a method by Dritschel [17] which was used previously for computing one-dimensional snapshots of invariant manifolds in 2D time-dependent flows (see [51, 50]). We therefore present only the main steps of the algorithm outlining the differences between the 2D and the 3D implementation.

The first step of the procedure consists of identifying the boundary geometry and generating a discrete copy $\mathfrak{B}_{t_0}$ [15], according to the specified resolution which is controlled by the curvature-dependent boundary segment length

$$\Delta X = 1/\big(R_{bnd}\, f(\kappa_\gamma)\big), \tag{124}$$

where R_{bnd} is the resolution parameter of the boundary which is independent of the mesh resolution, R, of the two-dimensional manifold snapshot used in (64). The form of the monotonically increasing function f is the same as (66) except the values of the cut-off parameters κ_{min} and κ_{max} are usually chosen to be, respectively, smaller and larger than in §5.1.

If the initial manifold snapshot is given in the analytical form, so that its boundary can be represented in the parametric form, the discretisation procedure consists of the following steps:

(1) Find the largest curvature value, κ_γ^{max} (see (43)), on the boundary curve and determine the shortest target length of a boundary segment, ΔX_{min}, by substituting κ_γ^{max} into (124).

(2) Generate a 'dense' sampling of the boundary, given by an ordered sequence of points on the boundary $\{\mathfrak{b}^j\}_{j\in J}$, so that the maximum separation between the points on the boundary does not exceed $\Delta X = \delta_{thr}\Delta X_{min}$; the user-specified threshold value is usually set to $\delta_{thr} = 0.1$.

(3) Generate the discretisation which conforms to the required resolution by choosing a subsequence $\{\mathfrak{b}^z\}_{z\in Z\subset J}$ of the dense discretisation in the following way:

i) Choose an arbitrary point, say $\mathfrak{b}^1$, on the boundary as the first point of the desired discretisation and compute ΔX_{target} by evaluating (124) on $\kappa_\gamma(\mathfrak{b}^1)$. Next, march along the boundary until the approximate arc length, i.e. $\Delta X = \sum_{i=1}^{k} |\mathfrak{b}^i - \mathfrak{b}^{i+1}|$, exceeds ΔX_{target}. The target length is updated after every step taken along the boundary points in order to avoid inaccuracies which might occur when the desired resolution underresolves the geometry of the boundary.

ii) Find the first point, say $\mathfrak{b}^k, (k > 1)$, which is separated (in the arc length sense) by ΔX_{target} from $\mathfrak{b}^1$. Add $\mathfrak{b}^k$ to the curvature based discretisation.

iii) Go to i) with $\mathfrak{b}^k$ as the initial point.

If the initial boundary is given as a discrete set of points (and provided that the resolution of the boundary is sufficiently high), the discretisation method used is essentially identical to step (3) in the algorithm outlined below. In both these cases a cubic interpolation method is used to approximate locations of new points on the boundary. The interpolation method is based on the one used by Dritschel in [17] but, due to the three-dimensionality of the problem considered here, there are significant differences in the formulae which we present below.

[15] The procedure of obtaining $\{\mathfrak{L}_{t_n}\}_{n\in Z_N}$ is identical.

Consider two adjacent points on the boundary, say, $\mathfrak{b}^i$ and $\mathfrak{b}^{i+1}$ and assume that between these points the boundary curve takes the form of a cubic spline, i.e.

$$\boldsymbol{x}_i(p) = \mathfrak{b}^i + p\boldsymbol{t}^i - \boldsymbol{\eta}^i(p), \quad 0 \leqslant p \leqslant 1, \tag{125}$$

where

$$\boldsymbol{t}^i = \mathfrak{b}^{i+1} - \mathfrak{b}^i, \tag{126}$$

and

$$\boldsymbol{\eta}^i(p) = \boldsymbol{\alpha}_i p + \boldsymbol{\beta}_i p^2 + \boldsymbol{\gamma}_i p^3. \tag{127}$$

The coefficients of the cubic interpolation vectors $\boldsymbol{\alpha}_i = (\alpha_x^i, \alpha_y^i, \alpha_z^i)^T$, $\boldsymbol{\beta}_i = (\beta_x^i, \beta_y^i, \beta_z^i)^T$, $\boldsymbol{\gamma}_i = (\gamma_x^i, \gamma_y^i, \gamma_z^i)^T$ are given by

$$\left.\begin{aligned} \alpha_x^i &= -\tfrac{1}{6}(\chi_x^{i+1} + 2\chi_x^i)\, d_x^i \\ \beta_x^i &= \tfrac{1}{2}\chi_x^i\, d_x^i, \\ \gamma_x^i &= \tfrac{1}{6}(\chi_x^{i+1} - \chi_x^i)\, d_x^i. \end{aligned}\right\} \tag{128}$$

where χ_x^i is given by

$$\chi_x^i = 2\frac{\langle \boldsymbol{t}^{i-1} - \boldsymbol{t}^i, \boldsymbol{e}_x\rangle}{|(d_x^{i-1})^2\boldsymbol{t}_x^i + (d_x^i)^2\boldsymbol{t}_x^{i-1}|}, \tag{129}$$

and

$$\boldsymbol{t}_x^i = \big(\langle \boldsymbol{t}^i, \boldsymbol{e}_x\rangle, 1\big)^T, \qquad d_x^i = |\boldsymbol{t}_x^i|. \tag{130}$$

The remaining coefficients are defined in an analogous way.

It can be verified that the polynomials (125) with the coefficients (128) satisfy $\boldsymbol{x}_i(0) = \mathfrak{b}^i$, $\boldsymbol{x}_i(1) = \mathfrak{b}^{i+1}$ and that they are C^1 at the boundary points. Therefore, in order to estimate curvatures at a point $\mathfrak{b}^i \in \mathfrak{B}_{t_n}$ of the boundary we take (cf. (43))

$$\kappa_\gamma^i \equiv \kappa_\gamma(\mathfrak{b}^i) \approx \frac{|\dot{\boldsymbol{x}}_i(0) \times (\ddot{\boldsymbol{x}}_{i-1}(1) + \ddot{\boldsymbol{x}}_i(0))|}{2|\dot{\boldsymbol{x}}_i(0)|^3}. \tag{131}$$

After discretising the initial boundary, $\mathfrak{B}_{t_0}$, the following steps are performed iteratively for $n \in Z_{N-1} = 0, \ldots, N-1$:

(1) Generate the boundary $\mathfrak{B}_{t_{n+1}}$ by evolving the location of the points in $\mathfrak{B}_{t_n}$ to the next observation time t_{n+1}. This step is performed in exactly the same way as for the interior nodes in the mesh $(\mathfrak{M}_{t_n}, \mathbb{M}^{t_n})$, i.e. by advecting the points of $\mathfrak{B}_{t_n}$ along trajectories of the system (2). Numerically, this reduces to integrating the equations (2) between t_n and t_{n+1} with the initial conditions given by the points in $\mathfrak{B}_{t_n}$.

(2) Identify boundary segments $\overline{\mathfrak{b}^i\mathfrak{b}^{i+1}}$ of $\mathfrak{B}_{t_{n+1}}$ which need to be refined. The identification of the 'long' segments is based on the value of the parameter σ^i, which is a function of the curvature κ_γ^i at the node (see (131)) and the separation to its nearest neighbours, i.e.

$$\sigma^i = \tfrac{1}{2} f(\kappa_\gamma^i)\,(d^{i-1} + d^i), \tag{132}$$

where $d^i = |\mathfrak{b}_i - \mathfrak{b}_{i+1}|$ and $f(\kappa_\gamma^i)$ has the form (66).

(3) Refine the 'long' boundary segments of $\mathfrak{B}_{t_{n+1}}$, if necessary. The parameter σ^i can be regarded as a fractional number of nodes to be placed between $\mathfrak{b}^i$ and $\mathfrak{b}^{i+1}$. If $\sigma^i > 1$ more resolution is needed, while for $\sigma^i < 1$ less resolution is required. If a segment is identified (assume $\overline{\mathfrak{b}^i\mathfrak{b}^{i+1}}$) for which $\sigma^i > 1$, a new node is inserted within the segment at the previous time t_n at the location given by $\boldsymbol{x}_i(0.5)$; this newly inserted point is later advected to t_{n+1}. The steps (2) and (3) are repeated until no long segments are left in $\mathfrak{B}_{t_{n+1}}$.

(4) Redistribute nodes within $\mathfrak{B}_{t_{n+1}}$. This process begins by fixing one node, say $\mathfrak{b}^1$, and identifying the node $\mathfrak{b}^k$ which precedes $\mathfrak{b}^1$ on the closed contour. Next, compute $q = \sum_{i=1}^k \sigma_i$ and define $\tilde{k} = [q] + 2$ (the nearest integer to q plus two). The redistribution procedure replaces the $k-1$ old nodes by $\tilde{k} - 1$ entirely new nodes in such a way that the spacing of the new nodes is approximately consistent with the desired average density

(see [17] for more details). If we now let $\sigma_i' = \sigma_i \tilde{n}/q$ so that $\sum_{i=1}^{n} = \tilde{n}$, the positions of new nodes are found successively by seeking i and p which satisfy

$$\sum_{l=1}^{i-1} \sigma_l' + \sigma_i' \, p = j - 1, \tag{133}$$

and then placing a new node between the old nodes $\mathfrak{b}^i$ and $\mathfrak{b}^{i+1}$ at the location $\boldsymbol{x}_i(p)$ given by (125). The accuracy of the redistribution is the same as the accuracy of the interpolation.

(5) Unless $n = n_{max}$, **go to (1)** with $n = n + 1$.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, *Computing and rendering point set surfaces*, IEEE Trans. Vis. Comput. Graph. **9** (2003), no. 1, 3–15.

[2] P. Alliez, M. Meyer, and M. Desbrun, *Interactive geometry remeshing*, ACM T. Graphic., Special Issue for SIGGRAPH conference **20(3)** (2002), 347–354.

[3] I. Babuska and A. K. Aziz, *On the angle criterion in the finite element method*, SIAM J. Numer. Anal. **13** (1976), 214–226.

[4] G.K. Batchelor, *An Introduction to Fluid Dynamics*, Cambridge University Press, 1999.

[5] D. Beige, A. Leonard, and S. Wiggins, *Invariant Manifold Templates for Chaotic Advection*, Chaos Solitons & Fractals **4** (1994), no. 6, 749–868.

[6] M. Brady, A. Leonard, and D. I. Pullin, *Regularized Vortex Sheet Evolution in Three Dimensions*, J. Comput. Phys. **146** (1998), 520–545.

[7] M. Branicki, A.M. Mancho, and S. Wiggins, *A Lagrangian description of transport associated with a Front-Eddy interaction: application to data from the North-Western Mediterranean Sea*, Physica D (submitted).

[8] J.H.E. Cartwright, M. Feingold, and O. Piro, *Chaotic advection in three-dimensional unsteady incompressible laminar flow*, J. Fluid Mech. **316** (1996), 259–284.

[9] J.C. Cavendish, *Automatic triangulation of arbitrary planar domains for the finite element method*, Int. J. Num. Meth. Eng. **8** (1974), 679–696.

[10] X. Chen and F. Schmitt, *Intrinsic surface properties from surface triangulation*, The Second European Conference on Computer Vision, 1992, pp. 739–743.

[11] S.S. Chern, W.H. Chen, and K.S. Lam, *Lectures on Differential Geometry*, Series on University Mathematics, World Scientific, 1999.

[12] W. A. Coppel, *Dichotomies in Stability Theory*, Lecture Notes in Mathematics, vol. 629, Springer-Verlag, New York, Heidelberg, Berlin, 1978.

[13] C. Coulliette and S. Wiggins, *Intergyre transport in a wind-driven, quasigeostrophic double gyre: An application of lobe dynamics*, Nonlinear Proc. Geoph. **8** (2001), 69–94.

[14] L. Dieci, R.D. Russell, and E.S. van Vleck, *On computation of Lyapunov exponents for continuous dynamical systems*, SIAM J. Numer. Anal. **34** (1997), 402–423.

[15] M. DoCarmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, NJ, 1976.

[16] E. J. Doedel, B. Krauskopf, and H. M. Osinga, *Global bifurcations of the Lorenz manifold*, Nonlinearity **19** (2006), no. 12, 2947–2972.

[17] D.G. Dritschel, *Contour dynamics and contour surgery - numerical algorithms for extended, high-resolution modeling of vortex dynamics in two-dimensional, inviscid, incompressible flows*, Comput. Phys. Rep. **10** (1989), no. 3, 77–146.

[18] N. Dyn, K. Hormann, S. Kim, and D. Levin, *Optimising 3D triangulations using discrete curvature analysis*, Mathematical Methods in CAGD: Oslo 2000, (Eds.) T. Lyche, L.L. Schumaker, Vanderbilt University Press, Nashville, TN, 2001.

[19] J. P. England, B. Krauskopf, and H. M. Osinga, *Computing two-dimensional global invariant manifolds in slow-fast systems*, Int. J. Bifurcat. Chaos **17** (2007), no. 3, 805–822.

[20] S. Fleishman, D. Cohen-Or, and C. T. Silva, *Robust moving least-squares fitting with sharp features*, ACM Trans. Graph. **24** (2005), no. 3, 544–552.

[21] R. Franke and G. Nielson, *Smooth interpolation of large sets of scattered data*, Int. J. Numer. Meth. Eng. **15** (1980), no. 11, 1691–1704.

[22] W.H. Frey and D. A. Field, *Mesh relaxation: A new technique for improving triangulations*, Int. J. Num. Meth. Eng. **31** (1991), 1121–1133.

[23] A. Fukuyu, T. Ruzi, and A. Kanai, *The Response of Hill's Vortex to a Small Three Dimensional Disturbance*, J. Phys. Soc. Jpn. **63(2)** (1994), 510–527.

[24] M. Garland and P. S. Heckbert, *Surface Simplification Using Quadric Error Metrics*, Computer Graphics, Annual Conference Series **31** (1997), 209–216.

[25] M. Garland, A. Willmott, and P. S. Heckbert, *Hierarchical face clustering on polygonal surfaces*, I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics (New York, NY, USA), ACM, 2001, pp. 49–58.

[26] P. L. George and E. Seveno, *The advancing-front mesh generation methods revisited*, Int. J. Num. Meth. Eng. **37** (1994), 3605–3619.

[27] T. N. T. Goodman, *Local Derivative Estimation for Scattered Data Interpolation*, Appl. Math. Comput. **68** (1995), 41–50.

[28] A. Gray, *Modern Differential Geometry of Curves and Surfaces, 2nd ed. Boca Raton*, CRC Press, 1993.

[29] B. Hamann, *Curvature approximation for triangulated surfaces*, Geometric modelling, Springer-Verlag, London, UK, 1993.

[30] D. Henry, *Geometric theory of semilinear parabolic equations, Lecture Notes in Mathematics*, vol. 840, Springer-Verlag: New York, Heidelberg, Berlin, 1981.

[31] Q. Huang, S. Flöry, N. Gelfand, M. Hofer, and H. Pottmann, *Reassembling fractured objects by geometric matching*, ACM Trans. Graph. **25** (2006), no. 3, 569–578.

[32] K. Ide, D. Small, and S. Wiggins, *Distinguished hyperbolic trajectories in time-dependent fluid flows: analytical and computational approach for velocity fields defined as data sets*, Nonlinear Proc. Geoph. **9** (2002), 237–263.

[33] Y. Ito and K. Nakahashi, *Surface Triangulation for Polygonal Models Based on CAD data*, Int. J. Numer. Methods Fluids **39**, no. 1.

[34] ———, *Direct Surface Triangulation Using Stereolithography Data*, AIAA Journal **40** (2002), no. 3, 490–496.

[35] N. Ju, D. Small, and S. Wiggins, *Existence and Computation of Hyperbolic Trajectories of Aperiodically Time-Dependent Vector Fields and Their Approximations*, Int. J. Bif. Chaos **13** (2003), 1449–1457.

[36] N. Ju and S. Wiggins, *On roughness of Exponential Dichotomy*, J. Math. Anal. Appl. **262** (2001), 39–49.

[37] S. J. Kim, C. H. Kim, and D. Levin, *Surface simplification using discrete curvature norm*, The third Israel-Korea Binational Conference on Geometric Modeling and Computer Graphics, 2001.

[38] R. Kolluri, *Provably good moving least squares*, ACM Trans. Algorithms **4** (2008), no. 2, 1–25.

[39] B. Krauskopf, H.M. Osinga, E.J. Doedel, M.E. Henderson, J. Guckenheimer, A. Vladimirsky, M. Dellnitz, and O. Junge, *A survey of method's for computing (un)stable manifolds of vector fields*, Int. J. Bifurcat. Chaos **15** (2005), no. 3, 763–791.

[40] P. Krsek, C. Lukács, and R. R. Martin, *Algorithms for computing curvatures from range data, A. Ball et al. (Eds.)*, The Mathematics of Surfaces VIII, Information Geometers, 1998.

[41] J. A. Langa, J. C. Robinson, and A. Suárez, *Stability, instability, and bifurcation phenomena in non-autonomous differential equations*, Nonlinearity **15** (2002), 887–903.

[42] D. Lazzaro and L. B. Montefusco, *Radial basis functions for the multivariate interpolation of large scattered data sets*, J. Comput. Appl. Math. **140** (2002), no. 1-2, 521–536.

[43] J. M. Lee, *Introduction to Topological Manifolds*, Springer, 2000.

[44] F. Lekien, S. C. Shadden, and J. Marsden, *Lagrangian coherent structures in n-dimensional systems*, J. Math. Phys. **48** (2007), no. 6, 065404.

[45] L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler, *A Local/Global Approach to Mesh Parameterization*, Eurographics Symposium on Geometry Processing, Copenhagen, Denmark, 2008.

[46] S. H. Lo, *A new mesh generation scheme for arbitrary planar domains*, Int. J. Num. Meth. Eng. **21** (1985), 1403–1426.

[47] E. Magid, O. Soldea, and E. Rivlin, *A comparison of Gaussian and mean curvature estimation methods on triangular meshes of range image data*, Comput. Vis. Imag. Und. **107** (2007), no. 3, 139–159.

[48] A. M. Mancho, E. Hernández-García, D. Small, and S. Wiggins, *Lagrangian transport through an ocean front in the North-Western Mediterranean Sea*, J. Phys. Oceanogr. **38** (2008), 1222–1237.

[49] A. M. Mancho, D. Small, and S. Wiggins, *Computation of Hyperbolic Trajectories and their Stable and Unstable Manifolds for Oceanographic Flows Represented as Data Sets*, Nonlinear Proc. Geoph. **11** (2004), 17–33.

[50] ———, *A tutorial on dynamical systems concepts applied to Lagrangian transport in oceanic flows defined as finite time data sets: Theoretical and computational issues* , Phys. Rep. **437** (2006), 55–124.

[51] A. M. Mancho, D. Small, S. Wiggins, and K. Ide, *Computation of Stable and Unstable Manifolds of Hyperbolic Trajectories in Two-Dimensional, Aperiodically Time-Dependent Vector Fields*, Physica D **182** (2003), 188–222.

[52] B. Mederos, L. Velho, and L.H. De Figueiredo, *Moving least squares multiresolution surface approximation*, Computer Graphics and Image Processing, 2003. SIBGRAPI 2003. XVI Brazilian Symposium on (2003), 19–26.

[53] D. S. Meek and D. J. Walton, *On surface normal and gaussian curvature approximations given data sampled from a smooth surface*, Comput. Aided Geom. D. **17** (2000), no. 6, 521–543.

[54] H. K. Moffatt and D. W. Moore, *The response of Hill's spherical vortex to a small axisymmetric disturbance*, J. Fluid Mech. **87** (1978), 749–760.

[55] J. S. Muldowney, *Dichotomies and asymptotic behaviour for linear differential systems*, Trans. A.M.S. **283(2)** (1984), 465–484.

[56] K. Nakahashi and D. Sharov, *Direct Surface Triangulation Using the Advancing Front Method*, 12th AIAA Computational Fluid Dynamics Conference (Collection of Technical Papers. Pt. 1 (A95-36501 09-34), Washington, DC, American Institute of Aeronautics and Astronautics), 1995, pp. 442–451.

[57] J. Ottino, *The Kinematics of Mixing: Stretching, Chaos, and Transport*, Cambridge University Press, Cambridge, 1989.

[58] M. Pauly, M. Gross, and L. P. Kobbelt, *Efficient simplification of point-sampled surfaces*, VIS '02: Proceedings of the conference on Visualization '02 (Washington, DC, USA), IEEE Computer Society, 2002, pp. 163–170.

[59] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross, *Shape modeling with point-sampled geometry*, ACM T. Graphic. **22** (2003), no. 3, 641–650.

[60] T. Rozi, *Evolution of the Surface of Hill's Vortex Subjected to a Small Three-Dimensional Disturbancefor the Cases of* $m = 0, 2, 3$*, and* 4, J. Phys. Soc. Jpn. **68 (9)** (1999), 2940–2955.

[61] D. Rypl and Z. Bittnar, *Triangulation of 3D surfaces: from parametric to discrete surfaces*, ICECT'03: Proceedings of the third international conference on Engineering Computational Technology (Edinburgh, UK), Civil-Comp Press, 2002, pp. 33–34.

[62] R. Samelson and S. Wiggins, *Lagrangian Transport in Geophysical Jets and Waves: The Dynamical Systems Approach*, Springer-Verlag, New York, 2006.

[63] P. Sander, S. Gortler, J. Snyder, and H. Hope, *Signal-specialised parameterisation*, Proc. 13th Eurographics Workshop on Rendering, 2002, pp. 87–100.

[64] G. R. Sell, *Non-autonomous differential equations and dynamical systems*, Trans. Am. Math. Soc. **127** (1967), 241–283.

[65] E. Seveno, *Towards an adaptive advancing front method*, Proc. 6th International Meshing Roundtable, Sandia National Laboratories, 1997, pp. 349–362.

[66] E. Shaffer and M. Garland, *Efficient adaptive simplification of massive meshes*, VIS '01: Proceedings of the conference on Visualization '01 (Washington, DC, USA), IEEE Computer Society, 2001, pp. 127–134.

[67] D. Shephard, *A two-dimensional interpolation function for irregularly spaced data*, Proceedings of the 23rd National Conference, ACM, 1968, pp. 517–523.

[68] D. Struik, *Lectures on Classical Differential Geometry*, Addison-Wesley Series in Mathematics, Addison-Wesley, 1961.

[69] V. Surazhsky and C. Gotsman, *Explicit surface remeshing*, SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing (Aire-la-Ville, Switzerland, Switzerland), Eurographics Association, 2003, pp. 20–30.

[70] ———, *High quality compatible triangulations*, Eng. Comput. **20** (2004), 147–156.

[71] S. Wiggins, *Chaotic Transport in Dynamical Systems*, Springer-Verlag, New York, 1992.

[72] ———, *Introduction to Applied Nonlinear Dynamical Systems and Chaos* , Springer, 2nd ed., New York, 2003.

[73] ———, *The Dynamical Systems Approach to Lagrangian Transport in Oceanic Flows*, Annu. Rev. Fluid Mech. **37** (2005), 295–328.